Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Iztok Oder

Lokalizacija mobilnega robota s pomočjo večsmerne kamere

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Danijel Skočaj

Ljubljana 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Lokalizacija je ena izmed temeljnih nalog v mobilni robotiki. Za uspešno ciljno usmerjeno navigacijo mora biti mobilni robot sposoben avtonomno ugotoviti svojo pozicijo v prostoru. V ta namen se najpogosteje uporabljajo podatki zajeti z globinskimi senzorji. V diplomski nalogi preučite drugi pristop, ki temelji na procesiranju slik zajetimi z večsmerno kamero. Na osnovi tako dobljenih panoramskih slik zgradite zemljevid okolice, ki bo omogočal avtonomno lokalizacijo robota zgolj na osnovi zajete panoramske slike. Za modeliranje okolja uporabite podprostorske metode kot sta Analiza glavnih komponent in Analiza kanonične korelacije. Implementirane metode aplicirajte na mobilnem robotu ATRV Mini. Razviti sistem naj omogoča navigacijo robota po prostoru ter avtonomno lokalizacijo temelječo na panoramskih slikah. Z robotom izvedite tudi eksperimentalno evaluacijo ter primerjajte natančnost implementiranih metod za lokalizacijo.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Iztok Oder, z vpisno številko **63110328**, sem avtor diplomskega dela z naslovom:

Lokalizacija mobilnega robota s pomočjo večsmerne kamere

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Danijela Skočaja,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 12. septembra 2014 Podpis avtorja:

Zahvaljujem se svojemu mentorju doc. dr. Danijelu Skočaju za vso strokovno pomoč, nasvete in usmeritve pri izdelavi diplomske naloge.

Prav tako se zahvaljujem Jaki Cikaču in Anžetu Rezelju za vso pomoč pri namestitvi senzorjev na robota.

Prav posebna zahvala gre sošolcem, s katerimi smo skupaj gulili šolske klopi, delali različne projekte in se učili za izpite in kolokvije zadnja tri leta. Hvala Milutin Spasić, Ernest Beličič, Žiga Lesar in Matic Tribušon za vso pomoč in deljenje znanja, ki sta me pripeljala tako daleč.

Na koncu bi se rad zahvalil tudi staršem, ki me že od otroštva spodbujata pri učenju in podpirata moralno in finančno.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
GPS	global positioning system	globalni sistem pozicioniranja
SfM	structure from motion	struktura iz gibanja
LIDAR	light detection and ranging	svetlobna detekcija in merjenje razdalje
RADAR	radio detection and ranging	radijska detekcija in merjenje razdalje
ROS	robot operating system	robotski operacijski sistem
PCA	principal component analysis	analiza glavnih komponent
KPCA	kernel principal component analysis	analiza glavnih komponent z jedri
CCA	canonical correlation analysis	analiza kanonične korelacije
KCCA	kernel canonical correlation analysis	analiza kanonične korelacije z jedri
BSD	Berkley software distribution	Berkleyeva programska distribucija
ZPR	zero phase representation	predstavitev z ničelno fazo

SVD	singular value decomposition	dekompozicija na singularne vrednosti
RBF	radial basis function	radialno simetrična funkcija
SLAM	simultaneous localization and mapping	sočasna lokalizacija in mapiranje

Povzetek

Roboti bodo v prihodnosti v veliki meri zamenjali človeško delovno silo. Biti bodo morali čimbolj avtonomni, kar vključuje tudi samostojno navigacijo po prostoru. Za uspešno navigacijo je potrebno najprej poznati svojo pozicijo v svetu.

Obstaja veliko metod, ki se ukvarjajo z lokalizacijo. Običajno te metode uporabljajo podatke zajete z globinskimi senzorji. V diplomskem delu se osredotočimo na lokalizacijo s pomočjo panoramskih slik zajetih z večsmerno kamero. Lokalizacijo izvajamo s pomočjo statističnih metod PCA, KPCA, CCA in KCCA. Iz učnih slik s temi metodami izračunamo nizkodimenzionalne podprostore na katere jih potem tudi projeciramo. Tako dobimo model okolja, v katerem se robot nahaja. S tem modelom lahko napovemo predvidene lokacije testnih slik. Vse metode implementiramo za interaktivno lokalizacijo s pomočjo robota ATRV mini. Natančnost lokalizacije v delu tudi ovrednotimo in podamo nekaj predlogov za izboljšavo celotnega sistema.

Ključne besede: lokalizacija, panoramske slike, PCA, KPCA, CCA, KCCA, robot, robotski operacijski sistem, večsmerna kamera.

Abstract

In future, robots will largely replace human labour. They will have to be as autonomous as possible. This includes the ability to self-navigate through space. A successful navigation requires the knowledge of one's location in space.

Many methods that deal with the problem of self-localization exist. Usually these methods use data acquired with a depth sensor. In this thesis we explore the possibilities of self-localization using only panoramic images obtained with omnidirectional camera. Localization is performed using statistical methods PCA, KPCA, CCA and KCCA. These methods produce a low-dimensional subspace from high-dimensional input images. These images are than projected onto the subspace, which gives us an alternative representation of the environment that can be used for predicting the locations of test images. All methods are implemented for use with mobile robot ATRV mini. The accuracy of self-localization is evaluated and few suggestions for the improvement are proposed.

Keywords: self-localization, panoramic images, PCA, KPCA, CCA, KCCA, robot, robot operating system, omnidirectional camera.

Poglavje 1

Uvod

1.1 Motivacija

Robotski sistemi nas dandanes spremljajo že na vsakem koraku. Človek si že od nekdaj prizadeva opraviti dela karseda hitro in s čim manj napora. Robotska natančnost, hitrost, neutrudljivost in seveda cena na dolgi rok pa so glavni razlogi za njihovo uporabo. Zaradi tega so v marsikateri tovarni že v dobršni meri izpodrinili človeško delovno silo. V prihodnosti bodo roboti v veliki meri nadomestili ljudi, ki opravljajo fizična dela. Ljudje bodo samo nadzorovali njihovo delo in ukrepali v primeru napak.

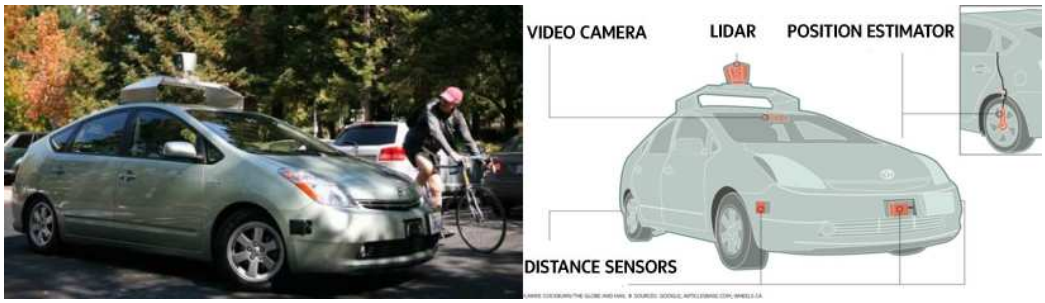
Minimalni posegi v robotovo delo so možni samo, če je robot karseda inteligenten in posledično avtonomen. V tovarni avtomobilov lahko naprimer vidimo, kako robotske roke samostojno sestavljajo avtomobile. Takšne roke so vnaprej sprogramirane za ponavljanje enega opravila. Temu ne moremo reči inteligentni sistem. Se pa med njimi najdejo tudi robotska vozila, ki so zadolžena za prevoz avtodelov iz enega dela tovarne v drugega. Sprogramirana so naprimer, da sledijo črtam različnih barv. Pri vožnji uporabljajo različne senzorje za zaznavanje ovir, s katerimi prilagajajo smer vožnje. Tak sistem je že inteligentnejši in posledično avtonomnejši.

Avtonomnost premikajočih robotov je mogoče doseči le z natančno lokalizacijo. Če robot ve, kje se v vsakem trenutku nahaja, lahko pride kamorkoli,

če je to fizično mogoče in če zna načrtovati pot od začetka do cilja mimo vseh ovir na poti. Lokalizacijo je mogoče doseči na več načinov.

Najbolj poznan je seveda globalni sistem pozicioniranja (ang. Global Positioning System) s kratico GPS. GPS uporablja štiri satelite, ki krožijo v orbiti zemlje, za izračun zemljepisne dolžine in širine ter nadmorske višine. Razlika, med časom sprejema signala in časom njegove oddaje, nam pove razdaljo med sprejemnikom in satelitom. Ko poznamo vse štiri razdalje do satelitov, je lokacijo sprejemnika enostavno določiti z matematičnimi enačbami [1].

Naslednja možnost je uporaba RADAR-ja (ang. RAdio Detection And Ranging). Ta sistem določa pozicijo objektov z oddajanjem radijskih valov in poslušanjem odbitkov. Iz analize prejetih valov lahko ugotovi obliko, smer potovanja, pozicijo in hitrost objektov. Na podoben način delujeta SONAR in LIDAR, pri čemer LIDAR uporablja drug del elektromagnetnega spektra, SONAR pa ultrazvok. Pozicija robota je v tem primeru izračunana glede na okolico [2]. Googlovo avtonomno vozilo naprimer uporablja LIDAR in RADAR za natančno lokalizacijo in izogibanje oviram in GPS globalno lokalizacijo. [3, 4]. Na sliki 1.1 so vidni Googlovo vozilo in senzorji, ki jih vozilo uporablja.



Slika 1.1: Googlovo avtonomno vozilo. Na desni sliki je prikazanih nekaj senzorjev. Povzeto po [5].

Ljudje takšnih senzorjev nimamo. Pri lokalizaciji se zanašamo predvsem na vizualne informacije. Vid je človekov najpomembnejši čut. Preko njega

dobimo največ informacij o okolju. Skupaj z možgani, ki te informacije obdelajo, lahko enostavno prepoznamo že videne stvari. Vizualna podoba sveta vsebuje ogromno informacij. Prepoznavanje ljudi in predmetov je za človeka enostavno opravilo. Prav tako je z lokalizacijo. Opravljamo jo brez posebnih težav, če smo okolico že videli. Ljudje si zapomnimo značilne karakteristike okolja v katerem se nahajamo. Zapomnimo si torej izstopajoče zgradbe, ceste, naravne pojave, itd. Na podlagi teh informacij, ob novem obisku iste lokacije, natančno vemo, kje se nahajamo.

Ta fenomen ni značilen samo za ljudi. Tudi živali z zgradbo telesa nakazujejo pomembnost vida. Marsikateri insekti, naprimer muhe, imajo oči prilagojene, da zajamejo čim večji vidni kot, kot se vidi na sliki 1.2. Vizualna informacija jim pomaga pri navigaciji in več informacij, kot jih prejmejo, tem lažje se orientirajo. Najbolj zanimivo je, da kljub majhni zmožnosti procesiranja izločijo najpomembnejše značilke potrebne za lokalizacijo in navigacijo.

Slika 1.2: Muha lahko s svojimi očmi zajame skoraj 360° vidni kot.

Ob upoštevanju zgornjih zmožnosti živalskih organizmov, lahko predvidevamo, da je možno bogatost vizualnih informacij izkoristiti tudi za lokalizacijo robota. Številne metode s področa računalniškega vida nam to omogočajo. Za uporabo teh metod potrebujemo le primerno poslikano okolje.

Okolje lahko poslikamo s kamero z navadno lečo. Rezultat so slike posnete pod največ 100° vidnim kotom. Vsak del okolice poslikamo iz več različnih lokacij in rotacij glede na horizontalno ravnino. Tako dobimo množico slik,

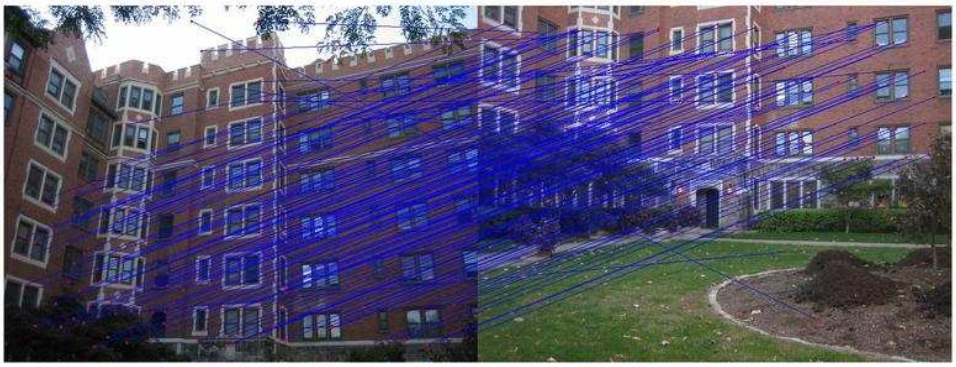
kjer se isti predmeti lahko pojavijo na večih slikah. Za relativno dobro predstavitev je potrebno zajeti zelo veliko število slik.

Bolj kompaktno predstavitev okolja nam dajo slike, ki so posnete pod večjim zornim kotom. Z njimi zajamemo večje področje, kar pomeni potrebo po manjšem številu slik. To se lahko doseže z različnimi sistemi kamer. Med take sisteme spadajo kamere s takoimenovanim objektivom ribje oko (ang. fisheye). Z njimi lahko enostavno zajamemo okolico pod 180° kotom. Slike posnete s takimi lečami so na prvi pogled popačene, kar pa se da z različnimi metodami popraviti. Večji zorni kot omogočajo tudi večsmerne kamere. Zrcalo, ki ga posname kamera, zajame točke pod 360° vidnim kotom. Take slike se da razviti v panoramo, kakršna je vidna na sliki 1.6. Napravi, s katerima lahko posnamemo takšne slike, sta vidni na sliki 1.3.

Slika 1.3: Levo: objektiv ribje oko. Desno: večsmerna kamera. Povzeto po [6, 7].

Lokalizacijo s pomočjo slik lahko naredimo tako, da vsako na novo pridobljeno sliko primerjamo s tistimi iz učne množice. Poiskati je potrebno značilke v učnih in testnih slikah (kot se vidi na sliki 1.4), ki se kar najbolj ujemajo [8, 9]. Velik odstotek enakih značilk bomo našli samo v učnih slikah, ki dejansko vsebujejo predmete iz testne slike. Ko v bazi najdemo slike, ki so kar najbolj podobne testni sliki, lahko pozicijo izračunamo relativno na

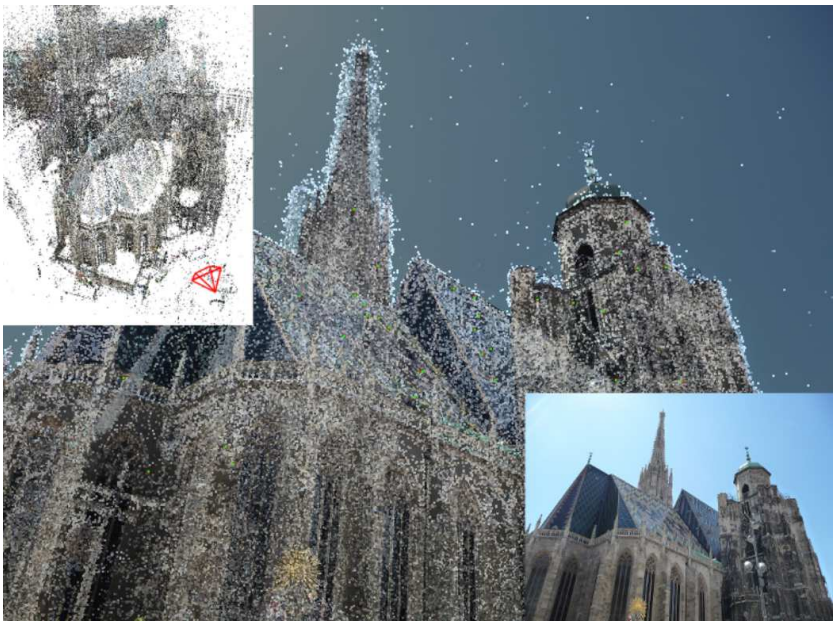
njihove pozicije. Taka lokalizacija je lahko samo tako dobra, kot je natančen GPS, s katerim smo dobili pozicije učnih slik [10, 11].



Slika 1.4: Značilke, ki povezujejo sliki.

Rekonstrukcija strukture iz gibanja (ang. Structure from Motion Reconstruction) je naslednji pristop, ki omogoča lokalizacijo. S tem pristopom je mogoče narediti 3D predstavitev okolice iz učnih slik. Za 2D točke na slikah je potrebno najti njihovo 3D predstavitev. Takšno rekonstrukcijo se da, zaradi napredka na področju SfM, narediti relativno hitro za scene velikosti celega mesta. Takšna predstavitev je precej prostorsko zahtevna, saj 3D scena vsebuje ogromno število točk. Problem se pojavi ob lokalizaciji nove slike. Zaradi kompleksnosti scene je iskanje značilk dolgotrajno [12, 13]. Na sliki 1.5 se vidi kompleksnost rekonstrukcije ene izmed vseh stavb v 3D prostoru. Na sliki je samo ena stavba in že tu se vidi ogromno število točk potrebnih za rekonstrukcijo. Za rekonstrukcijo celotnega mesta je število teh toč zelo veliko, kar pa upočasnjuje iskanje značilk.

Panoramske slike (primer panorame je viden na sliki 1.6) so se prav tako dobro izkazale pri lokalizaciji. Predstaviti se jih da namreč v nizkodimenzionalnih prostorih. Lokalizacija temelji na primerjanju projekcije nove slike v tem prostoru z projekcijami učnih slik. Najbolj pogosta metoda, ki se uporablja za tako predstavitev je PCA (ang. Principal Component Analysis) [14], obstajajo pa tudi druge [15].



Slika 1.5: 3D rekonstrukcija zgradbe. Povzeto po [13].

Slika 1.6: Panoramska slika.

Lokalizacija samo s pomočjo vizualne informacije je zagotovo mogoča. V tej diplomski nalogi se osredotočimo na lokalizacijo s pomočjo panoramskih slik.

1.2 Cilji diplomske naloge

Glavni cilj je implementacija sistema za lokalizacijo s pomočjo panoramskih slik zajetih z večsmerno kamero. Lokalizacija temelji na predstavitvi panoramskih slik v podprostorih. Metode uporabljene za ta namen so PCA (ang. Principal Component Analysis), KPCA (ang. Kernel Principal Component Analysis), CCA (ang. Canonical Correlation Analysis) in KCCA (ang. Kernel Canonical Correlation Analysis). Vse metode implementiramo za uporabo na mobilnem robotu ATRV mini. Robot omogoča samostojno navigacijo z uporabo globinskega senzorja Kinect in lokalizacijo, ki temelji na panoramskih slikah. Metode tudi evalviramo in primerjamo njihovo natančnost.

1.3 Zgradba diplomske naloge

Diplomska naloga je razdeljena na pet poglavij. V prvem poglavju naredimo kratek uvod v tematiko, s katero se ukvarja diplomska naloga in definiramo cilje. Drugo poglavje poda opis principa lokalizacije, učnih metod in priprave panoramskih slik. V tretjem poglavju predstavimo sestavne dele strojne in programske opreme in opišemo delovanje implementacije lokalizacijskega sistema. Predzadnje, četrto poglavje, je namenjeno predstavitvi in diskusiji rezultatov lokalizacije in implementiranega sistema. Zadnje poglavje predstavlja zaključek z nekaj predlogi izboljšav.

Poglavje 2

Teoretična podlaga

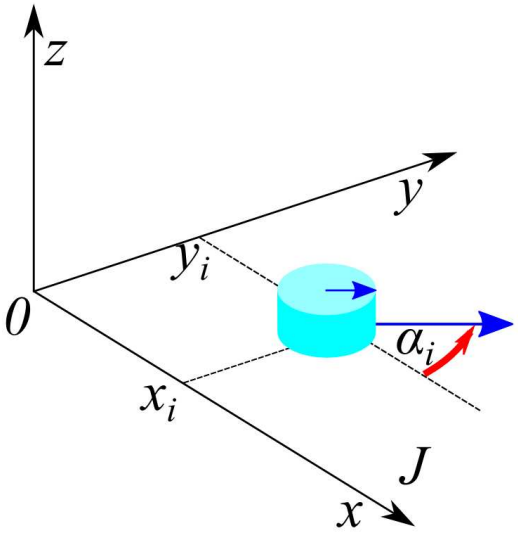
2.1 Strojno učenje

Strojno učenje je področje umetne inteligence, ki se ukvarja s pridobivanjem znanja na podlagi izkušenj. Učenje dejansko predstavlja iskanje pravil in vzorcev v učnih podatkih.

Pri problemu lokalizacije robota iščemo vzorce v panoramskih slikah posnetih v nekem prostoru. To so visokodimenzijski podatki, ki jih dobimo s pomočjo večsmerne kamere. Slike, ki pri snemanju prostora nastanejo, delimo na dve množici.

Prva je učna množica. To je množica slik z oznakami (x, y, α). Oznake so koordinate (x_i, y_i) v prostoru, kjer je bila slika posneta, in rotacija kamere α_i v ravnini J določeni z osema x in y, kot je vidno na sliki 2.1. Slike z oznakami nam dajo predstavitev prostora.

Iz učnih slik je potrebno zgraditi klasifikacijski ali regresijski model. Naloga modela je posploševanje znanja, ki ga dobi iz učne množice. Model predstavlja zveze med učnimi slikami in oznakami, ki jih definirajo slike. Oznake predstavljajo razrede, kamor lahko katerokoli novo sliko klasificiramo, če uporabljamo klasifikator. Regresijski model ni omejen samo na zalogo vrednosti, ki jo predstavljajo oznake učnih slik. Vhodne podatke lahko preslika na zvezno področje vrednosti. Izhod obeh modelov je predvidena lokacija nove slike



Slika 2.1: Oznake v 3D prostoru.

(x_j, y_j) in kot rotacije α_j v ravnini J.

Druga množica so takoimenovane testne slike. Teh slik ni v učni množici. Uporabljamo jih za preverjanje natančnosti modelov za klasifikacijo in regresijo. Prav tako so opremljene z oznakami (x, y, α), vendar se le-te uporabljajo izključno za preverjanje natančnosti.

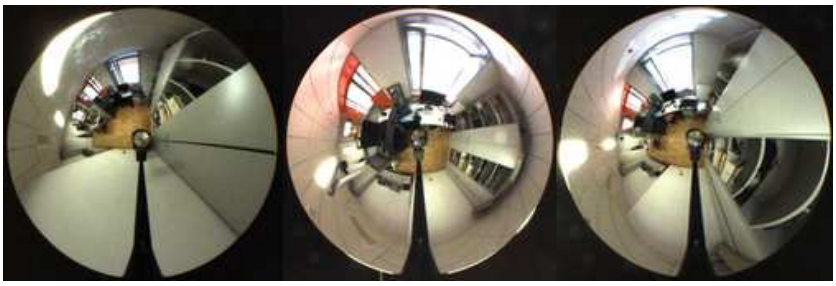
2.2 Priprava podatkov

2.2.1 Transformacija v panoramsko sliko

Večsmerna kamera za slikanje okolice, kot je predstavljena v poglavju 3, preslika točke iz okolice na sliko v polarnih koordinatah. Odvisno od namena uporabe se lahko takšne slike transformirajo v bolj razumljive slike. Z ustrezno preureditvijo slikovnih elementov originalne slike lahko iz človeku navidez nerazumljive slike dobimo panoramske ali pa perspektivne slike.

Na sliki 2.2 so prikazane originalne slike. Slike so na prvi pogled precej nerazumljive. Vidi se, da je posnet prostor, vendar je dokaj težko ugotoviti, kaj vse se nahaja v prostoru. Panoramske slike, ki jih dobimo s transforma-

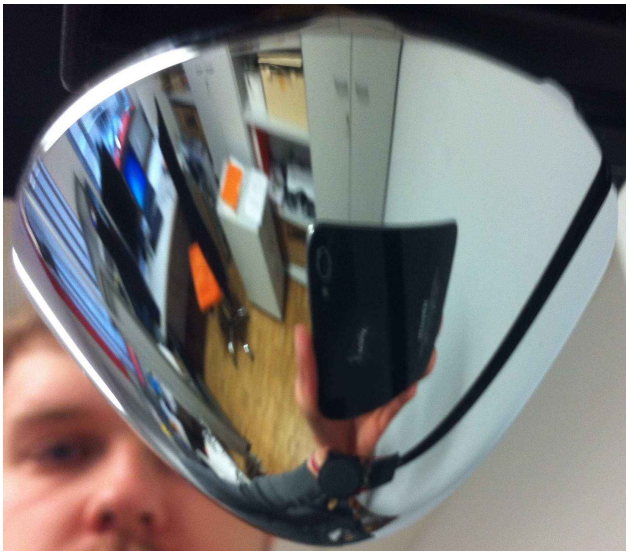
cijo, so vidne na sliki 2.3. Predmete v prostoru je v takšni sliki precej lažje razločiti. Za primerjavo je na sliki 2.4 prikazan odsev okolice v zrcalu iz malo drugačne perspektive.



Slika 2.2: Originalne slike.

Slika 2.3: Panoramske slike.

Panoramske slike, kot so vidne na sliki 2.3, dobimo s transformacijo slik iz polarnega koordinatnega sistema v kartezičnega. Metod za tako transformacijo je več. Najbolj enostavna je direktna transformacija iz enega sistema v drugega. Bolj kompleksne metode, ki med drugim odpravljajo tudi problem navpične popačenosti vsebine slike, uporabljajo lastnosti zrcala za bolj natančno transformacijo [16]. V našem sistemu smo uporabili kar prvo, enostavnejšo metodo.

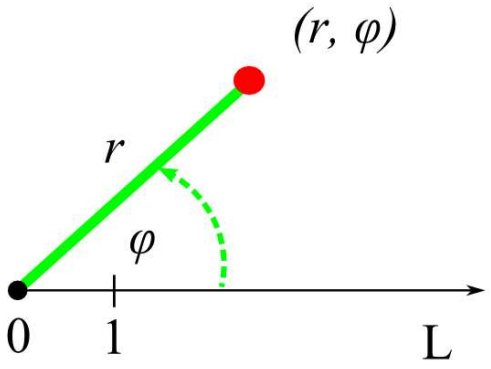


Slika 2.4: Okolica v zrcalu.

Točka je v polarnem koordinatnem sistemu predstavljena z dvema parametroma:

- radijem r, ki meri oddaljenost od izhodišča koordinatnega sistema in
- polarnim kotom φ, ki ga določa točka glede na desni del vodoravne osi.

Enolično določenost s parametroma se vidi na sliki 2.5. Rdeča pika predstavlja točko v koordinatnem sistemu.



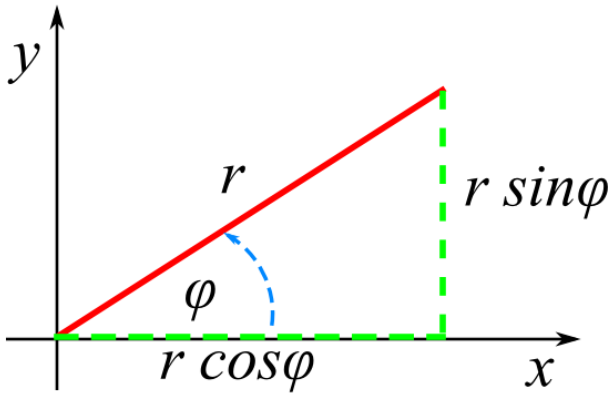
Slika 2.5: Polarne koordinate.

Za tako predstavljeno točko v polarnem koordinatnem sistemu je enostavno najti predstavitev v kartezičnem koordinatnem sistemu z naslednjima enačbama:

$$x = r * cos(\varphi) \tag{2.1}$$

$$y = r * sin(\varphi). \tag{2.2}$$

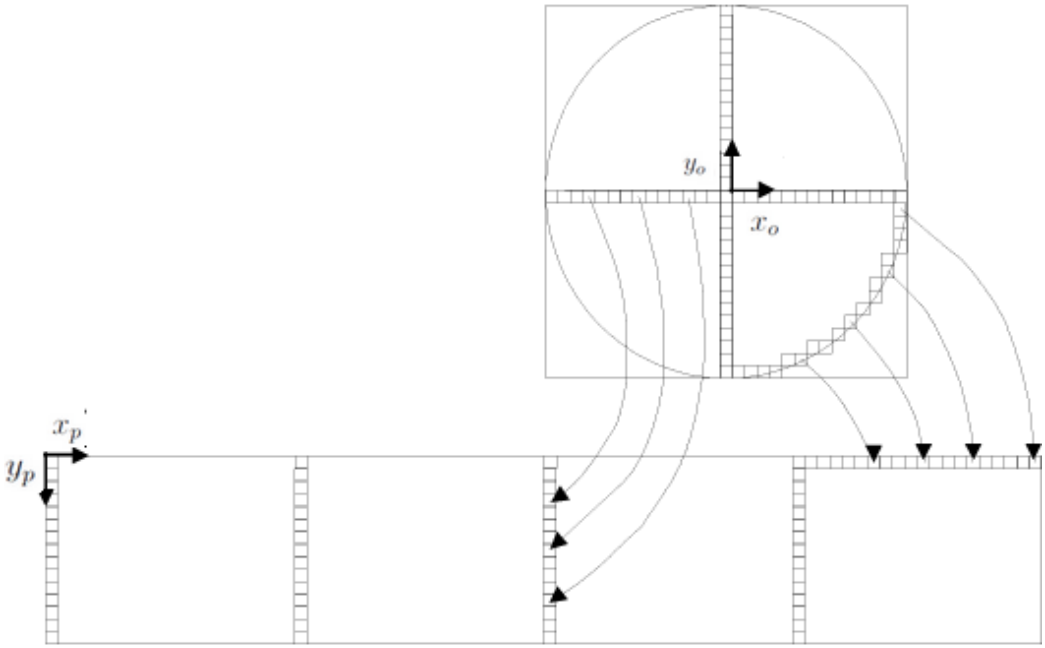
To zvezo med polarnimi in kartezičnimi točkami, kot jo opisujeta enačbi (2.1) in (2.2), se vidi na sliki 2.6.



Slika 2.6: Polarne koordinate v kartezičnem koordinatnem sistemu.

Transformacijo originalne slike v panoramsko dobimo tako, da polarni kot originalne slike pretvorimo v vodoravno os panoramske slike in radij originalne slike v navpično os panoramske slike, kot to prikazuje slika 2.7.

V panoramski sliki je število slikovnih elementov v vseh vrsticah slike enako. Pri pretvarjanju originalne slike v panoramsko, se slikovni elementi na krožnici, ki je določena z r_o, prepišejo v najvišjo vrstico panoramske slike, slikovni elementi na krožnici določeni z r_i, pa se prepišejo v najnižjo vrstico. Dolžina krožnic je različna, kar lahko povzroči, da pri predvorbi nekateri slikovni elementi, ki so bližje spodnji vrstici, ostanejo prazni. Zato se transformacija izvaja v obratni smeri. Za vsak slikovni element v panoramski sliki poiščemo ustrezen slikovni element v originalni sliki z enačbami (2.3) - (2.6). Tako lahko zagotovimo, da se bo en slikovni element iz originalne slike pre-



Slika 2.7: Pretvorba med slikama. Povzeto po [17].

slikal v točno določen slikovni element iz panoramske slike. Hkrati bodo vsi slikovni elementi panoramske slike zapolnjeni [17].

$$\varphi = \frac{2 * \pi * x_p}{width} \tag{2.3}$$

$$r = r_i + \frac{(height - y_p) * (r_o - r_i)}{height} \tag{2.4}$$

$$x_o = r * cos(\varphi) + x_c \tag{2.5}$$

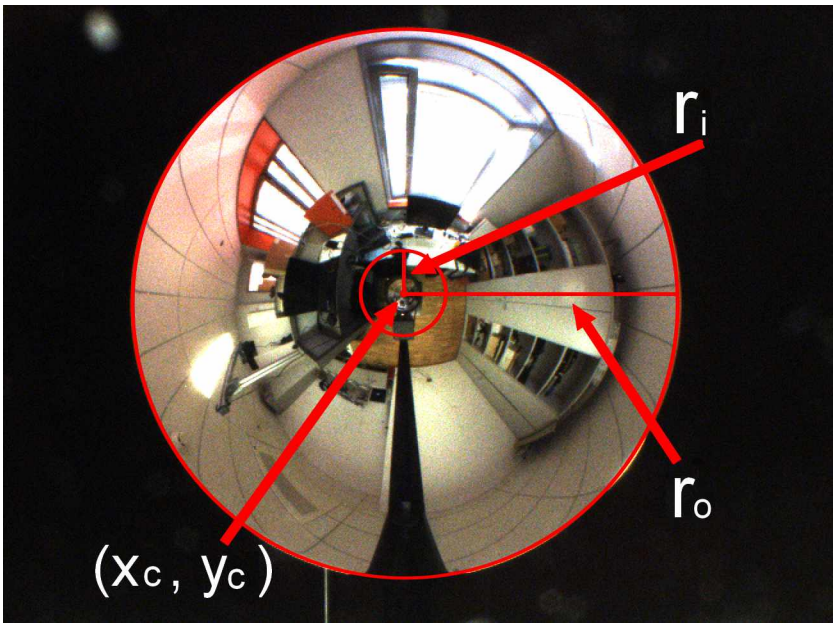
$$y_o = r * sin(\varphi) + y_c \tag{2.6}$$

Parametri zgornjih enačb so razloženi v tabeli 2.1 in predstavljeni na slikah 2.7 in 2.8.

Takšno enostavno vzorčenje slikovnih elementov v originalni sliki lahko povzroči, da je pridobljena panoramska slika slabše kvalitete. To lahko dodatno izboljšamo z vzorčenjem več sosednjih slikovnih elementov in na njih izvedemo bilinearno ali bikubično interpolacijo.

oznaka	pomen
φ	polarni kot točke v originalni sliki
r	radij točke v originalni sliki
x_o	koordinata x slikovnega elementa v originalni sliki
y_o	koordinata y slikovnega elementa v originalni sliki
x_c	koordinata x centra originalne slike
y_c	koordinata y centra originalne slike
r_i	notrani polmer v originalni sliki
r_o	zunanji polmer v originalni sliki
x_p	koordinata x slikovnega elementa v panoramski sliki
y_p	koordinata y slikovnega elementa v panoramski sliki
$height$	višina panoramske slike
$width$	širina panoramske slike

Tabela 2.1: Tabela prikazuje parametre uporabljene v enačbah (2.3) - (2.6).



Slika 2.8: Parametri za transformacijo.

2.2.2 Izravnava histograma

Pri lokalizaciji s panoramskimi slikami naletimo na problem različne osvetljenosti slik. Pri uporabi katerekoli izmed statističnih metod opisanih v tem poglavju se lahko zgodi, da se slika posneta na isti lokaciji narobe preslika v nižjedimenzijski prostor. Da omilimo ta problem, je potrebno vsako panoramsko sliko popraviti z neko metodo za odstranitev premajhne ali prevelike osvetljenosti in izboljšanje globalnega kontrasta.

Slika 2.9 prikazuje različno osvetljene slike iste scene. V levem stolpcu so originalne panoramske slike. Opazna je razlika v osvetljenosti. Na slikah stoji človek. V prvih dveh se ga še vidi, v tretji pa je skoraj neopazen. V desnem stolpcu so popravljene različice istih slik. Globalna osvetljenost slik je v tem stolpcu zelo podobna.

Slika 2.9: Levi stolpec: originalne slike. Desni stolpec: slike popravljene z izravnavo histograma.

Izravnava histograma je tehnika za izboljševanje globalnega kontrasta. Na ta način je mogoče omiliti vpliv premajhne ali prevelike osvetljenosti slike. V sivinski sliki vsak slikovni element zavzema neko vrednost sivine na lestvici, kjer najmanjše možno število predstavlja črno barvo in največje število belo. Zaloga vrednosti nivojev sivine je omejena na vrednosti med 0 in $L - 1$. L predstavlja število možnih vrednosti nivojev sivine in ponavadi zavzema vrednost 256. Ob zajemu sivinske slike je lahko zaradi različnih pogojev v okolju in nastavitev kamere zaloga vrednosti nivojev sivine precej

bolj skrčena in zavzema le manjši del celega intervala. Take slike imajo majhen kontrast. Izravnava histograma omogoči razširitev tega intervala na celoten možen interval zaloge vrednosti, kot je to vidno na histogramih na sliki 2.10. Na sliki je vidna tudi sprememba v osvetljenosti in kontrastu po pretvorbi iz originala.

Naj x predstavlja sivinsko sliko velikosti $p \times q$. Vsak slikovni element na sliki zavzema vrednost med 0 in $L-1$. Verjetnost pojavitve slikovnega elementa z nivojem sivine x_i je predstavljena z $P(x_i)$. Torej

$$P(x_i) = \frac{n_i}{n}, \qquad i = 0, 1, 2, .., L-1, \tag{2.7}$$

pri čemer $n = p * q$ predstavlja število vseh slikovnih elementov in n_i število slikovnih elementov z nivojem sivine enakim x_i. Vrednosti $P(x_i)$ predstavljo slikin histogram normaliziran na interval $[0, 1]$.

Izravnavo histograma dosežemo z uporabo transformacijske funkcije

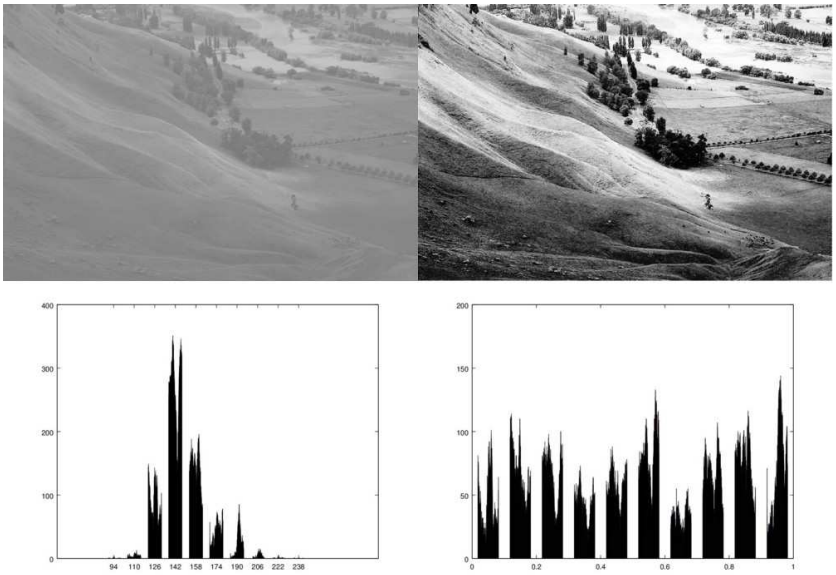
$$T(x_i) = \sum_{j=0}^{i} P(x_j) = \sum_{j=0}^{i} \frac{n_j}{n}, \qquad i = 0, 1, 2, \ldots, L-1. \tag{2.8}$$

Sprocesirano sliko dobimo tako, da vsak slikovni element nivojem sivine x_i iz originalne slike, pretvorimo s pomočjo transformacijske funkcije v slikovni element z novim nivojem sivine $y_i = T(x_i)$ [18].

2.3 Učne metode

2.3.1 Analiza glavnih komponent

Metoda analize glavnih komponent (ang. Principal Component Analysis, PCA) je statistična metoda, ki je našla svoje mesto v računalniškem vidu. Najprepoznavnejši aplikaciji te metode sta prepoznavanje obrazov in kompresija slik. Čeprav formulacijo metode pripisujejo H. Hotellingu v delu [20], začetke uporabe metode lahko zasledimo že leta 1901 v delu K. Pearsona [21]. Metoda je standardna tehnika za iskanje vzorcev v visokodimenzijskih

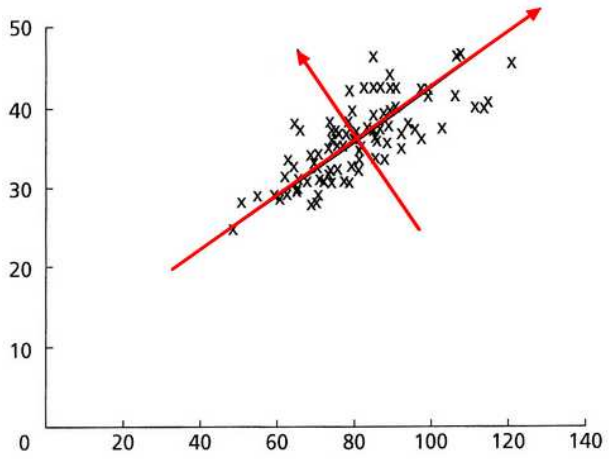


Slika 2.10: Levo zgoraj: slika s slabim kontrastom. Levo spodaj: histogram slike s slabim kontrastom. Desno zgoraj: popravljena slika z izravnavo histograma. Desno spodaj: histogram popravljene slike. Povzeto po [19].

podatkih in predstavitvi podatkov v obliki, kjer so njihove podobnosti in razlike opazne [22].

PCA naredi linearno transformacijo visokodimenzionalnih podatkov iz originalnega prostora v nov nizkodimenzionalni podprostor, ki najbolje predstavi variacijo podatkov od povprečja. Ta nov prostor je predstavljen z baznimi vektorji, ki razpenjajo nov koordinatni sistem. Na sliki 2.11 so prikazani dvodimenzionalni podatki. PCA je našel nov podprostor razpet z baznimi vektorji označenimi z rdečimi puščicami.

PCA maksimizira varianco projekcij podatkov v podprostor. Variance podatkov v smeri posameznega vektorja so različne. Največ variance zajema vektor, ki predstavlja prvo glavno komponento koordinatnega sistema. Drugi vektor zajema že manj variance, tretji še manj in tako naprej. Zadnji vektorji zajemajo zelo malo variance in jih za predstavitev podatkov v koordinatnem sistemu ne potrebujemo, saj jo preostali vektorji že dovolj dobro opišejo. PCA minimizira rekonstrukcijsko napako med originalnimi projekcijami in



Slika 2.11: Podatki v originalnem sistemu. Na sliki je viden tudi nov koordinatni sistem. Povzeto po [23].

projekcijami pri uporabi manjšega števila vektorjev [24].

Slika 2.12 predstavlja podatke iz slike 2.11 projecirane v nov podprostor razpet z dvema baznima vektorjema. Bazna vektorja sta izbrana tako, da je rekonstrukcijska napaka med projekcijami na vse vektorje in projekcijami na manjše število baznih vektorjev minimalna. Zaradi preglednosti je minimizacija prikazana samo za dve projekciji iz množice vseh. Napaka je označena z zeleno črto. Projekcija na dva bazna vektorja je označena z modro piko, projekcija na en bazni vektor pa z rdečo piko. Iz slike 2.12 je lahko razbrati, da je varianca v smeri prvega vektorja večja kot v smeri drugega.

Izračun baznih vektorjev

V naši učni množici imamo N slik velikosti $p \times q$, ki jih predstavimo kot stolpične vektorje $x_i \in \mathbb{R}^{n\times 1}$, $n = p * q$. Vse slike zložimo v matriko $X = [x_0, x_1, x_2, \ldots, x_{N-1}]$, $X \in \mathbb{R}^{n\times N}$. To matriko je potrebno centrirati okoli središča trenutnega koordinatnega sistema. To bo povzročilo, da bodo podatki tudi v novem koordinatnem sistemu centrirani okoli središča. Vsaki sliki je potrebno odšteti povprečno sliko, torej vsakemu stolpcu matrike X

Slika 2.12: Podatki predstavljeni v novem koordinatnem sistemu. Povzeto po [23].

je potrebno odšteti povprečni stolpec

$$\mu_x = \frac{1}{N}\sum_{i=0}^{N-1} x_i, \qquad \mu_x \in \mathbb{R}^{n\times 1}. \tag{2.9}$$

Standardni način za izračun baznih vektorjev novega koordinatnega sistema je rešitev problema lastnih vrednosti

$$Cu_i = \lambda_i u_i, \tag{2.10}$$

kjer λ predstavlja lastne vrednosti, u pa pripadajoče lastne vektorje matrike C. Matrika C je kovariančna matrika

$$C = \frac{1}{N} XX^\top, \qquad C \in \mathbb{R}^{n\times n}. \tag{2.11}$$

Največja lastna vrednost predstavlja največjo varianco, pripadajoči lastni vektor pa pove smer z največjo varianco.

Enačbo (2.10) je možno rešiti z diagonalizacijo matrike C. To napravimo z metodo SVD *(ang. Singular Value Decomposition)* in dobimo

$$C = ULU^\top, \tag{2.12}$$

kjer $L \in \mathbb{R}^{N\times N}$ vsebuje lastne vrednosti na diagonali in $U \in \mathbb{R}^{n\times N}$ predstavlja matriko pripadajočih lastnih vektorjev.

Ta način je, ker imamo opravka z slikami, računsko in prostorsko izjemno zahteven. Zato je potrebna malo drugačna formulacija enačb. Namesto diagonalizacije kovariančne matrike naredimo diagonalizacijo notranjega produkta matrike X:

$$O = \frac{1}{N} X^\top X, \qquad O \in \mathbb{R}^{N \times N}, \tag{2.13}$$

ki ima precej manjše dimenzije kot kovariančna, saj velja $N << n$. Lastne vrednosti matrike O so enake lastnim vrednostim kovariančne matrike, lastne vektorje kovariančne matrike, pa lahko izračunamo s pomočjo lastnih vektorjev matrike O in enačbe:

$$u_i = \frac{X u_i{}'}{\sqrt{N * \lambda_i{}'}}, \qquad i = 0, 1, 2, ..., N-1. \tag{2.14}$$

Projeciranje in rekonstrukcija

S pomočjo izračunanih lastnih vektorjev, ki predstavljajo bazne vektorje novega koordinatnega sistema, lahko preslikamo podatke v nov podprostor. Pri tem obdržimo samo podmnožico vektorjev, ki pripadajo največjim lastnim vrednostim. Število vektorjev k lahko določimo s pomočjo energije, ki jo vsebujejo lastni vektorji. Energija prvih k vektorjev se izračuna po enačbi:

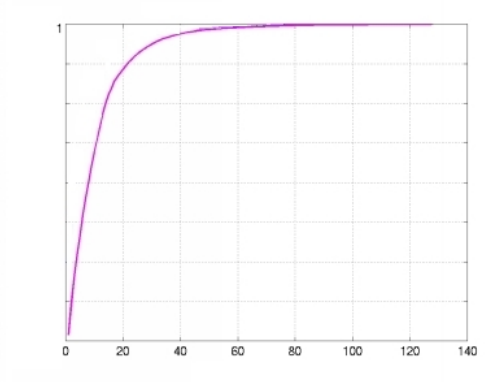
$$en = \frac{\sum_{i=0}^{k-1} \lambda_i}{\sum_{i=0}^{N-1} \lambda_i}. \tag{2.15}$$

Na sliki 2.13 se vidi, da prvih nekaj vektorjev vsebuje največ energije. Ostali vektorji doprinesejo zanemarljivo malo energije, zato jih ne potrebujemo.

Ko določimo število k, s transformacijsko matriko $U_k \in \mathbb{R}^{n \times k}$, ki vsebuje prvih k lastnih vektorjev, projeciramo sliko x v podprostor in dobimo njene koeficiente.

$$c = U_k^\top (x - \mu_x) \tag{2.16}$$

V primeru projeciranja panoramskih slik se lahko zgodi, da se zaradi nepredvidenih ovir v sliki, le-ta projecira v prostor na drugačno mesto od



Slika 2.13: Energija baznih vektorjev.

pričakovanega. Takšne ovire se pojavijo v sliki zaradi dinamičnosti okolja, v katerem so bile slikane. V prostoru se lahko sprehajajo osebe, na zrcalu večsmernega sistema oziroma kameri se lahko pojavi kakšna smet, itd. V tem primeru se testna slika precej razlikuje od učnih, ki so pridobljene v bolj ali manj kontroliranih pogojih. Zgornja metoda projeciranja je nerobustna, ker upošteva vse slikovne elemente, in zato nenatančna. Napako pri računanju koeficientov lahko odpravimo z robustno metodo predlagano v članku [25].

Namesto uporabe vseh slikovnih elementov, se omejimo le na l, $k < l << n$, slikovnih elementov, ki jih naključno izberemo iz množice vseh slikovnih elementov. Koeficiente c lahko izračunamo kot rešitev predeterminiranega sistema linearnih enačb. Če označimo naključni indeks vrstice vektorja slike x (x je že centriran) z r_i, $r \in \mathbb{R}^l$, lahko nastavimo sistem enačb:

$$x_{r_i} = \sum_{j=0}^{k-1} U_{k_{r_i,j}} c_j, \qquad i = 0, 1, 2, \ldots, l-1, \tag{2.17}$$

kjer x_{r_i} predstavlja vrednost vektorja x v vrstici r_i, $U_{k_{r_i,j}}$ predstavlja vrednost matrike U_k v vrstici r_i in stolpcu j in c_j predstavlja vrednost j-tega koeficienta.

Rešitev c dobimo po metodi najmanjših kvadratov, ki minimizira napako

$$E(r) = \sum_{i=0}^{l-1}(x_{r_i} - \sum_{j=0}^{k-1} U_{k_{r_i,j}} c_j)^2. \tag{2.18}$$

Seveda se lahko zgodi, da je znotraj množice slikovnih elementov x_r kakšnen osamelec, ki zaradi svoje vsebine zelo prispeva k napaki. Zato je potrebno iz množice slikovnih elementov x_r iterativno odstranjevati tiste, ki največ prispevajo k napaki, in narediti ponovni izračun koeficientov (2.17) in napake (2.18). To ponavljamo dokler ni napaka dovolj majhna $E(r) \leq \theta$,

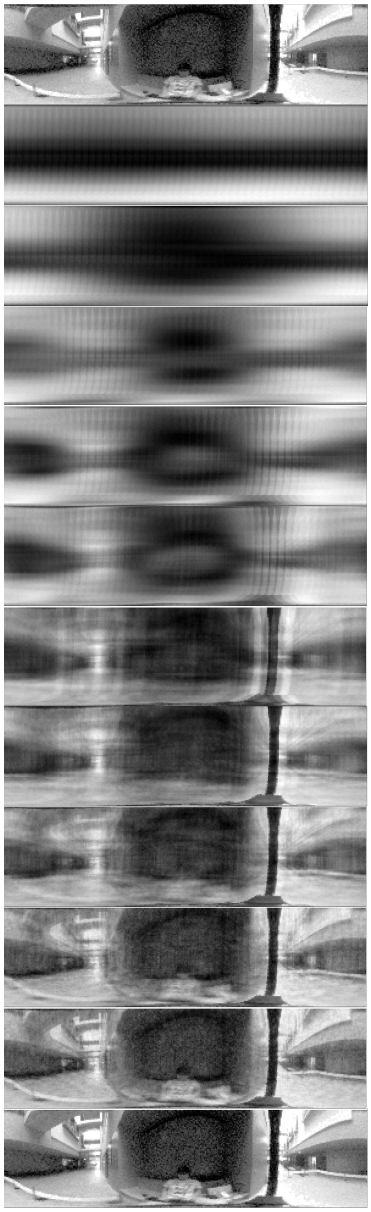
$$\theta = \frac{2}{n} \sum_{i=k}^{N-1} \lambda_i, \tag{2.19}$$

oziroma dokler je število slikovnih elementov, uporabljenih za računanje koeficientov, dovolj veliko. Da povečamo verjetnost pravilno izbranih slikovnih elementov, celoten postopek izbire koeficientov nekajkrat ponovimo. Iz te množice hipotez, ki jih predstavljajo vektorji koeficientov, napaka rekonstrukcije in število slikovnih elementov, iz katerih smo izračunali koeficiente, nato po principu najkrajšega opisa izberemo najboljšo [25].

Pridobljene koeficiente je mogoče projecirati nazaj v originalen prostor z enako transformacijsko matriko U_k in uporabo enačbe (2.20). Pridobljena rekonstruirana slika je linearna kombinacija lastnih vektorjev v matriki U_k [26].

$$y = U_k c + \mu_x \tag{2.20}$$

Primer rekonstrukcije je viden na sliki 2.14. Prva slika predstavlja original. Vse ostale so produkt rekonstrukcije projekcij pri različnem številu baznih vektorjev. Število uporabljenih baznih vektorjev za posamezno rekonstrukcijo si sledi: 1, 2, 5, 10 , 20, 50, 100, 200, 500, 800, 1200. Vidi se, kako število baznih vektorjev vpliva na kakovost rekonstrukcije. Pri uporabi enega baznega vektorja slika ni niti malo podobna originalu. Z večanjem števila vektorjev pridobivamo vedno boljšo rekonstrukcijo. Pri uporabi vseh baznih vektorjev dobimo original.

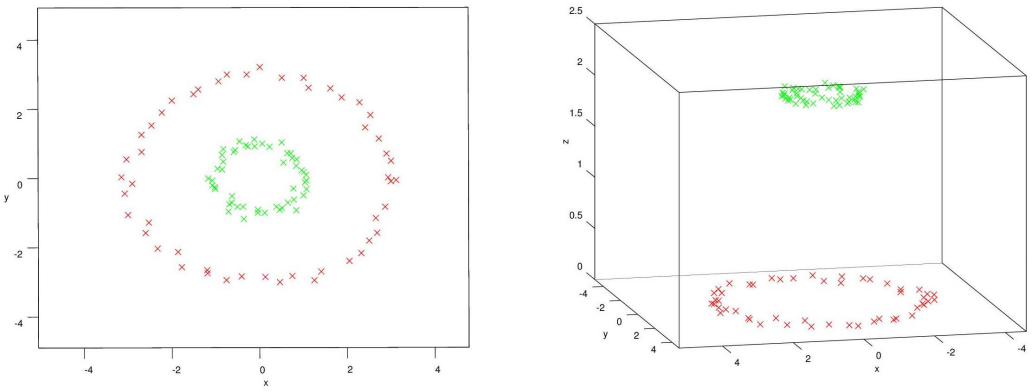


Slika 2.14: Rekonstrukcija slik pri različnem številu vektorjev.

2.3.2 Analiza glavnih komponent z jedri

KPCA (ang. Kernel Principal Component Analysis) uporablja praktično isto idejo kot analiza glavnih komponet. Z njo želimo projecirati neke podatke iz visokonivojskega prostora, kjer relacije med njimi niso opazne, v nek nov

nizkonivojski podprostor. Podprostor mora tako kot pri PCA zajemati čim več variance v podatkih. Pomembna razlika med PCA in KPCA je to, da PCA naredi linearno transformacijo podatkov, medtem ko KPCA ne. KPCA najprej pretvori podatke v nek nov visokonivojski prostor F s transformacijsko funkcijo Φ in na teh podatkih naredi linearno transformacijo v nov podprostor. Tako dosežemo nelinearno transformacijo originalnih podatkov v podprostor. Slika 2.15 prikazuje podatke v 2D prostoru pred transformacijo in podatke v 3D prostoru po transformaciji. Dodatna dimenzija razkrije možnost linearne razdružitve podatkov.



Slika 2.15: Primer 2-dimenzijskih originalnih podatkov in njihova predstavitev po transformaciji. Levo: podatki v 2D prostoru. Desno: podatki v 3D prostoru. Po transformaciji je opazna linearna razdružitev podatkov.

Pretvorba podatkov s transformacijsko funkcijo Φ je izjemno časovno in prostorsko zahtevna, oziroma v nekaterih primerih celo nemogoča. Dimenzija transformiranih podatkov je precej večja kot dimenzija originalnih podatkov, celo neskončna. Na srečo takšna pretvorba ni potrebna. Algoritmi, ki se jih lahko spremeni tako, da uporabljajo le skalarne produkte, lahko uporabijo takoimenovan trik z jedrom. Namesto pretvorbe podatkov v prostor F in nato računanja matrike notranjih produktov O, lahko uporabimo funkcijo imenovano Mercerjevo jedro $k(x, y)$. Dokazano je, da je ta funcija ekvivalentna skalarnemu produktu $\Phi(x)^\top * \Phi(y)$. S to funkcijo, lahko za vse podatke x_i

(v našem primeru slike), izračunamo matriko $K_{i,j} = k(x_i, x_j)$. Ta matrika je ekvivalentna notranjemu produktu matrike $\Phi(X)$, kjer stolpci matrike predstavljajo $\Phi(x_i)$ [27].

Obstaja veliko Mercerjevih jedr. Najbolj pogosto pa se pojavljajo gaussovo oziroma RBF *(ang. Radial Basis Function)* jedro (2.21), polinomsko jedro (2.22) in sigmoidno jedro (2.23).

$$k(x, y) = e^{\frac{-|x-y|^2}{2*\sigma^2}} \tag{2.21}$$

$$k(x, y) = (x^\top y + c)^d \tag{2.22}$$

$$k(x, y) = tanh(\alpha * x^\top y + c) \tag{2.23}$$

Pri jedrih je seveda izjemno pomembna izbira parametrov, ki jih lahko določimo kar eksperimentalno.

Izračun baznih vektorjev

Naj bo učna množica X sestavljena tako kot pri PCA. Od stolpcev je potrebno odšteti povprečni stolpec μ_x, da bodo podatki centrirani. Tako zagotovimo, da bodo po preslikavi s funkcijo Φ tudi podatki v tem prostoru centrirani

$$\sum_{i=0}^{N-1} \Phi(x_i) = 0. \tag{2.24}$$

Bazni vektorji so tako kot pri PCA povezani z rešitvijo splošnega problema lastnih vrednosti

$$K * u_i = N * \lambda_i * u_i, \tag{2.25}$$

zato je naslednji korak izbira jedra in izračun matrike K

$$K_{i,j} = \Phi(x_i)^\top * \Phi(x_j), \qquad K \in \mathbb{R}^{N \times N}. \tag{2.26}$$

Matrika K je simetrična, zato lahko čas računanja prepolovimo, saj velja $K_{i,j} = K_{j,i}$. To precej pospeši računanje relativno velike matrike.

V primeru, da učni podatki niso centrirani, je potrebno matriko K normalizirati

$$K = K - 2 * 1_{\frac{1}{N}} K + 1_{\frac{1}{N}} K 1_{\frac{1}{N}}, \tag{2.27}$$

kjer $1_{\frac{1}{N}}$ predstavlja matriko velikosti $N \times N$ z elementi $\frac{1}{N}$.

Lastne vektorje dobimo s pomočjo metode SVD, ki diagonalizira našo matriko K.

$$K = ULU^{\top} \tag{2.28}$$

V primeru, da je število učnih primerov N večje od velikosti n posamezne slike ($N > n$), lahko najdemo s PCA največ n neničelnih lastnih vrednosti. KPCA jih lahko najde N. Zaradi tega je manšanje števila dimenzij podprostora opcijsko [28].

Projekcija in rekonstrukcija

Projekcija v podprostor, razpet z vektorji v matriki U, prav tako ne potrebuje eksplicitnega računanja preslikave nove slike x v $\Phi(x)$. Projekcije na U lahko izračunamo preprosto z enačbo:

$$c = U^{\top}\Phi(x) = \sum_{i=0}^{N-1} u_i * k(x, x_i) \tag{2.29}$$

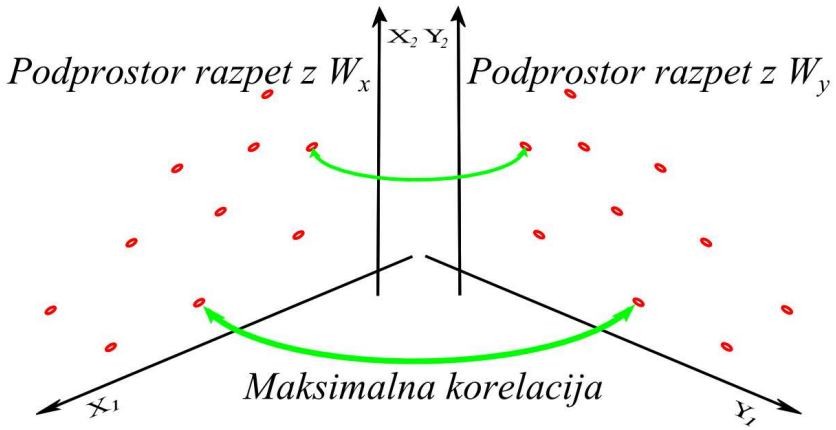
in tako dobimo koeficiente, ki predstavljajo projekcijo slike v podprostoru.

Pri PCA je bilo možno narediti inverzno projekcijo iz podprostora in tako dobiti originalne podatke oziroma njihove približke. Pri KPCA so rezultati v nekem višjedimenzijskem prostoru in zato ni nujno, da inverzne projekcije sploh obstajajo. Vseeno obstajajo metode, s katerimi lahko dobimo približke originalov [29].

2.3.3 Analiza kanonične korelacije

CCA (ang. Canonical Correlation Analysis) je statistična metoda preučevanja linearnih odvisnosti med dvema množicama podatkov, ki jo je leta 1936 predlagal Hotelling. To je metoda nadzorovanega učenja, saj učenje delamo na

dveh množicah. Prva množica je množica neodvisnih spremenljivk, druga pa množica odvisnih spremenljivk. Metoda poišče dve množici baznih vektorjev (za vsako originalno svojega), ki razpenjata dva podprostora. Korelacija med projekcijami podatkov iz prve množice na prvi podprostor in projekcijami druge množice podatkov na drugi podprostor je pri tem največja možna. To prikazuje slika 2.16. Število baznih vektorjev je omejeno z manjšo od obeh velikosti dimenzij posameznega učnega primera iz obeh množic [30, 31].



Slika 2.16: Maksimizacija korelacije med projekcijami.

Izračun baznih vektorjev

Pri CCA imamo namesto ene množice dve. Prva učna množica je množica N slik $X = [x_0, x_1, ..., x_{N-1}]$, $X \in \mathbb{R}^{n \times N}$, kjer stolpci predstavljajo raztegnjene slike. Druga množica so pozicije kamere, kjer so bile slike posnete. Vsak stolpec v tej množici $Y = [y_0, y_1, ..., y_{N-1}]$, $Y \in \mathbb{R}^{z \times N}$ predstavlja lokacijo v navideznem koordinatnem sistem in rotacijo kamere okoli svoje osi. Od vsakega primera v obeh množicah je potrebno odšteti povprečni stolpec μ_x oziroma μ_y in tako centrirati podatke.

CCA poišče bazne vektorje W_x in W_y dveh sistemov tako, da maksimizira korelacijo med projekcijami $W_x^\top X$ in $W_y^\top Y$. Število vektorjev je v splošnem omejeno z

$$min(n, z). \tag{2.30}$$

V našem primeru je minimalna vrednost vedno z. Vektorje lahko izračunamo z rešitvijo splošnega problema lastnih vrednosti

$$Aw_i = \lambda_i B w_i, \tag{2.31}$$

kjer velja

$$A = \begin{bmatrix} 0 & C_{xy} \\ C_{yx} & 0 \end{bmatrix}, \quad B = \begin{bmatrix} C_{xx} & 0 \\ 0 & C_{yy} \end{bmatrix}, \quad W = \begin{bmatrix} W_x \\ W_y \end{bmatrix} \tag{2.32}$$

in

$$\begin{aligned} C_{xx} &= \frac{1}{N} X X^\top, \quad & C_{yy} &= \frac{1}{N} Y Y^\top, \\ C_{xy} &= \frac{1}{N} X Y^\top, \quad & C_{yx} &= \frac{1}{N} Y X^\top. \end{aligned} \tag{2.33}$$

Takšna oblika CCA je seveda v našem primeru neprimerna. Velikosti slik so precej velike in zato je računanje kovariančne matrike C_{xx} izjemno računsko in prostorsko zahtevno. Zato je, tako kot pri PCA, potrebna malo drugačna oblika računanja W. Še vedno rešujemo problem (2.31), vendar namesto zgornjih matrik A in B vzamemo malo drugačni

$$A = \begin{bmatrix} 0 & OY^\top \\ YO & 0 \end{bmatrix}, \quad B = \begin{bmatrix} OO & 0 \\ 0 & C_{yy} \end{bmatrix}, \tag{2.34}$$

kjer je

$$O = \frac{1}{N} X^\top X. \tag{2.35}$$

Matrika notranjih produktov je po velikosti precej manjša od kovariančne, saj je običajno $N << n$. Podobno spremembo bi lahko naredili v primeru, da bi bile dimenzije C_{yy} prevelike.

Za rešitev enačbe (2.31) je potreben izračun inverza matrike B. Zaradi možnih numeričnih problemov je pomembno, da sta matriki C_{xx} in C_{yy}, ki sestavljata B, polnega ranga. V nasprotnem primeru pride do singularnosti matrike B. V izogib teh problemov je potrebna regularizacija te matrike. To naredimo tako, da matriki B prištejemo večkratnik identitete αI, $\alpha > 0$. Če je α dovolj velik, matrika B postane pozitivno definitna, kar omogoči

izračun njenega inverza. Z regularizacijo se ne izognemo samo numeričnim problemom, ampak tudi problemom prevelikega prilagajanja učnim podatkom [32, 33].

Zaradi drugačne oblike CCA je potrebno pridobljen W'_x pretvoriti v W_x, podobno kot pri PCA, s projeciranjem na X

$$Wx = XW'_x. \tag{2.36}$$

CCA je idealna za regresijske probleme, saj lahko izračunamo transformacijsko funkcijo, s katero lahko napovemo vrednost y kamor se preslika spremenljivka x. Transformacijska funkcija je definirana kot

$$F = YT^{\dagger}, \tag{2.37}$$

kjer C predstavlja matriko koeficientov učne množice projecirane na podprostor razpet z vektorji iz W_x, torej

$$T = W_x^{\top} X, \tag{2.38}$$

in $T^{\dagger}$ njen moore-penrosov inverz [15].

Za vsako novo sliko x lahko s pomočjo zgornje transfomacijske funkcije izračunamo njeno vrednost y po enačbi:

$$y = FW_x^{\top}(x - \mu_x) + \mu_y. \tag{2.39}$$

2.3.4 Analiza kanonične korelacije z jedri

KCCA (ang. Kernel Canonical Correlation Analysis) je nelinearna različica CCA. CCA je zmožna med dvema množicama podatkov ugotoviti samo linearne odvisnosti. Tako kot pri KPCA je možna transformacija obeh množic podatkov v nove višjedimezijske podatke. Prvo množico, množico neodvisnih spremenljivk, pretvorimo s funkcijo Φ, drugo, množico odvisnih, pa s Θ. Katerakoli od obeh funkcij je lahko enaka linearnemu jedru, ki producira matriko notranjih produktov originalnih podatkov. Tako je možno transformirati le množico neodvisnih spremenljivk in s tem ohraniti prednost CCA,

kjer lahko z linearno regresijo izračunamo odvisno spremenljivko po (2.39). Če transformiramo tudi množico odvisnih spremenljivk, je potrebno pri rezultatu pridobljenem z regresijo narediti inverzno preslikavo iz visokodimenzijskega prostora v originalen prostor odvisne spremenljivke. Taka preslikava pa ni nujno obstoječa. S KCCA je mogoče izračunati več baznih vektorjev, kot je omejitev pri navadni CCA.

Izračun baznih vektorjev

Izračun baznih vektorjev W_x in W_y poteka podobno kot pri CCA. Kot pri CCA imamo dve množici podatkov X in Y. Vsaki posamezni je potrebno odšteti ustrezen povprečni stolpec. Tako bosta obe po transformaciji z Φ in Θ v višjenivojskem prostoru centrirani okoli izhodišča.

Rešitev splošnega problema (2.31) ustreza baznim vektorjem v W_x in W_y, ki razpenjajo ustrezna podprostora. Pomembna razlika pri računanju je sestava matrik A in B. Sestavljeni sta iz matrik $K1$ in $K2$,

$$\begin{aligned} K1_{i,j} &= \Phi(x_i)^\top \Phi(x_j) \\ K2_{i,j} &= \Theta(x_i)^\top \Theta(x_j). \end{aligned} \tag{2.40}$$

Matriki A in B tako postaneta enaki

$$A = \begin{bmatrix} 0 & K1K2 \\ K2K1 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} K1K1 & 0 \\ 0 & K2K2 \end{bmatrix}. \tag{2.41}$$

Projekcija

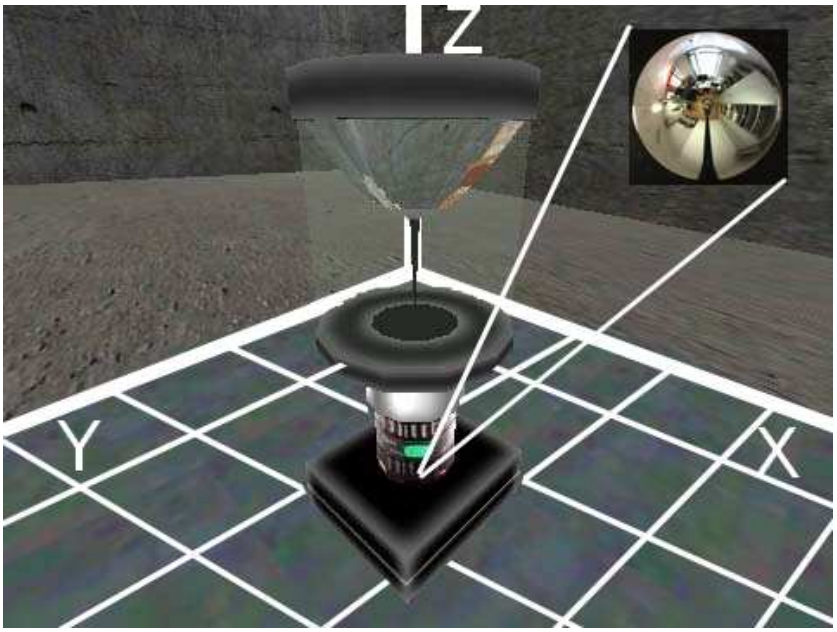
Projekcija v podprostor, razpet z vektorji v matriki W_x, prav tako ne potrebuje eksplicitnega računanja preslikave nove slike x v $\Phi(x)$. Projekcije na W_x lahko izračunamo preprosto z enačbo

$$c = W_x^\top \Phi(x) = \sum_{i=0}^{N-1} w_{x_i} * k(x, x_i) \tag{2.42}$$

in tako dobimo vektor s koeficienti, ki predstavlja podatke v podprostoru.

2.4 Lokalizacija s panoramskimi slikami

Učne slike je potrebno posneti na določenih mestih v prostoru, kjer se bo lokalizacija izvajala. Za vsako posneto sliko moramo poznati, koordinati x_i in y_i v navideznem koordinatnem sistemu znotraj prostora, kot to prikazuje slika 2.17. Poleg tega moramo poznati tudi kot α_i, za katerega je zavrtena večsmerna kamera v ravnini J določeni z osema x in y. Slike je potrebno posneti v čimvečih, karseda enakomerno razporejenih točkah, da čimbolj enakomerno zajamemo variabilnost prostora.



Slika 2.17: Zajemanje slike v 3D prostoru z večsmerno kamero. Povzeto po [34].

Učne slike nato z metodami PCA, KPCA, CCA ali KCCA, ki so opisane v poglavju 2.3, predstavimo v nekem manj kompleksnem podprostoru. To naredimo tako, da izračunamo bazne vektorje podprostorov, kot je opisano v poglavju 2.3. Na dobljene podprostore projeciramo učne slike in pridobimo njihove vektorje koeficientov. Pri CCA je potrebno izračunati tudi funkcijo za linearno regresijo, kot je opisano v poglavju 2.3.3.

Lokalizacija s CCA je zelo preprosta in hitra. Testno sliko je potrebno projecirati v podprostor, nato pa s pomočjo regresijske funkcije izračunati predvideno lokacijo po enačbi (2.39).

Drugače je pri uporabi podprostorov izračunanih s PCA, KPCA ali KCCA. Vsako testno sliko v fazi lokalizacije prav tako preslikamo v ustrezen podprostor in jo primerjamo z učnimi slikami v tem podprostoru. Primerjavo se naredi z računanjem evklidske razdalje med testno sliko in vsako posamezno učno sliko. Najmanjša razdalja predstavlja najbližjo učno sliko testni sliki v podprostoru. Lokacija najbližje učne slike predstavlja napovedano lokacijo testne slike.

Takšna lokalizacija je natančna samo toliko, kolikor so lokacije, kjer so bile učne slike posnete, razmaknjene ena od druge. Za natančnejšo lokalizacijo je potrebno vektorje koeficientov učnih slik interpolirati in tako zgenerirati vektorje koeficientov tudi na vmesnih lokacijah.

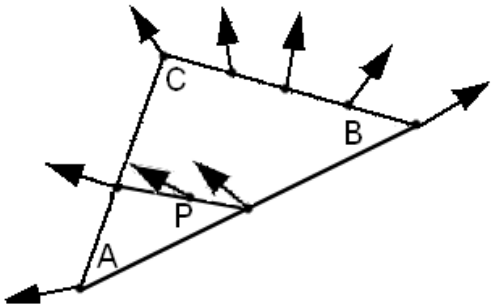
Koeficienti učnih slik predstavljajo polje vektorjev v ravnini J. Vektorji koeficientov so postavljeni v tistih točkah (x_i, y_i), kjer so bile originalne slike posnete. Tri sosednje vektorje lahko baricentrično interpoliramo in tako dobimo gostejše polje vektorjev koeficientov. Za interpolacijo vektorjev je potrebno najprej poznati baricentrične koordinate točk $P_i = (x_{P_i}, y_{P_i})$ znotraj trikotnika, določenega s točkami $A = (x_A, y_A)$, $B = (x_B, y_B)$ in $C = (x_C, y_C)$. Te točke predstavljajo lokacije, kjer so bile učne slike posnete. Baricentrične koordinate α, β in γ točke P_i so določene z enačbama:

$$P_i = A + \beta(B - A) + \gamma(C - A) \tag{2.43}$$

$$\alpha = 1 - \beta - \gamma. \tag{2.44}$$

Ko so baricentrične koordinate vseh točk P_i določene, lahko za vsako od njih izračunamo interpolirani vektor, kot to prikazuje slika 2.18. Na sliki se vidi, kako se smer in velikost vektorjev spreminja v odvisnosti od oddaljenosti od ogljišč trikotnika. Interpolirani vektorji predstavljajo približke projekcij slik, ki bi jih dobili s projeciranjem pravih slik, posnetih na istih lokacijah.

Po interpolaciji imamo precej več vektorjev koeficientov, s katerimi lahko



Slika 2.18: Baricentrična interpolacija vektorjev.

primerjamo vektorje koeficientov testne slike. Povečanje natančnosti lokalizacije je odvisno od gostote interpoliranih vektorjev.

Natančnost lokalizacije ni samo odvisna od razdalje med lokacijami, kjer so bile posnete učne slike. Lastnost panoramskih slik je, da imajo neglede na kot α, pod katerim so posnete, enako vsebino. Problem se pojavi, ko slike posnete na isti lokaciji vendar pod različnim kotom preslikamo v podprostor. Slike z enako vsebino se različno preslikajo v podprostor in posledično pride do napačne lokalizacije.

Zgornji problem je mogoče rešiti na več načinov. Ena možnost je, da vse slike, tako učne kot testne, zarotiramo v neko referenčno pozicijo. Opravka imamo s panoramskimi slikami, kar pomeni, da je rotacija slik preprost krožni zamik posameznih vrstic slike za nek kot α oziroma neko število stolpcev. Kako, oziroma za kakšen kot, je potrebno zarotirati lahko izvemo na več načinov.

1. Najenostavnejša je uporaba kompasa, če imamo do njega seveda dostop. Z njim enostavno ugotovimo za kakšen kot α je kamera zarotirana glede na neko referenčno lokacijo. Sliko enostavno odrotiramo za ugotovljen kot. Torej vsako posamezno vrstico krožno zamaknemo za

$$n = -\frac{\alpha * width}{2 * \pi} \tag{2.45}$$

stolpcev, pri čemer je $width$ širina slike.

2. Naslednja možnost je uporaba vsebine slike za ugotavljanje referenčne orientacije z metodo ZPR (ang. Zero Phase Representation)[35, 36]. Pri tej metodi postavimo sliko v neko referenčno rotacijo s postavitvijo faze prvega harmonika fourierjeve transformacije te slike na nič. Metoda je zelo občutljiva na spremembe v okolju.

Pri zgornjem načinu je bilo potrebno tako učne kot testne slike rotirati v pravilno pozicijo. Druga možnost odpravlja to pomankljivost. Namesto, da bi slike rotirali v referenčno pozicijo, vsako učno sliko rotiramo k-krat in tako iz vsake slike dobimo k slik. Kot rotacije za i-to sliko se izračuna po enačbi:

$$\alpha_i = \frac{2 * pi * i}{k}, \quad i = 0, 1, 2, ..., k - 1. \tag{2.46}$$

Iz N učnih slik tako dobimo $N * k$ slik. V podprostoru so predstavljene različno rotirane slike iz istih lokacij. Projekcija testne slike je v splošnem bolj podobna projekciji učne slike, ki je enako orientirana kot testna slika, kot pa projekcijam slik, ki so drugače orientirane [37, 14].

Slika 2.19: Zgoraj: originalni sliki. Spodaj: zarotirani tako, da je kot rotacije 0.

Poglavje 3

Implementacija

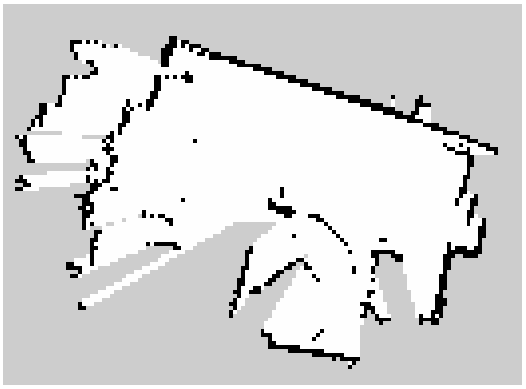
3.1 Oris robota in programske opreme

Za testiranje lokalizacijskih metod je potrebno imeti sistem, s katerim lahko slikamo prostor z večsmerno kamero. Robot, ki smo ga mi uporabili za ta namen, je bil ATRV mini. Na njega je bila nameščena konstrukcija za večsmerno kamero, kot je to vidno na sliki 3.1. Vsaka slika potrebuje tudi natančno lokacijo, kjer je bila posneta, in kot rotacije, pod katerim je bila posneta. Na sliki 3.3 je viden Kinect, ki je bil uporabljen za določanje referenčnih lokacij in rotacij.

Kinect se uporablja za snemanje območja, kjer želimo izvajati lokalizacijo, v povezavi s sistemom SLAM za grajenje zemljevidov zasedenosti. Zemljevid vsebuje ovire v prostoru, kot je to vidno na sliki 3.2. V istem prostoru posnamemo z večsmerno kamero učne slike. Za vsako sliko s pomočjo zemljevida zasedenosti pridobimo lokacijo in orientacijo robota, kjer je učna slika posneta. Enako pridobimo pozicijske podatke v fazi testiranja lokalizacijskih metod. Te podatke lahko nato primerjamo z napovedanimi.

Za upravljanje robota in senzorjev je bil uporabljen robotski operacijski sistem (ROS). V ROS-u je bil implementiran sistem za interaktivno lokalizacijo. Z njim smo implementirali dele programske opreme, ki so bili zadolženi za slikanje okolice z večsmerno kamero, navigacijo po prostoru, učenje na-

Slika 3.1: ATRV mini s Kinectom in z večsmerno kamero.



Slika 3.2: Primer posnete mape.

povednih modelov iz slik in napovedovanje pozicijskih podatkov na podlagi testnih slik.

3.2 ATRV mini

ATRV mini je izdelek podjetja iRobot, katerega proizvodnja je bila že ukinjena. Robot je precej masiven in zmogljiv. Namenjen je za delovanje v kateremkoli okolju. Ima pogon na štiri kolesa, ki jih lahko neodvisno upravljamo. S tem je omogočeno obračanje na mestu, ki je nadvse pomembno pri uporabi v manjših prostorih. Dimenzije robota so: $150cm \times 53cm \times 63cm$. Možgane robota predstavlja zelo zmogljiv računalnik s specifikacijami podanimi v tabeli 3.1. Nanj je mogoče priključiti različne senzorje, kot so GPS sprejemnik, pospeškometer, ultrazvočni senzor, različne kamere, itd. Za naše potrebe smo nanj priključili senzor Kinect in večsmerno kamero. Robot je prikazan na slikah 3.1 in 3.3, kjer sta vidna tudi oba senzorja.

3.3 Kinect

Kinect je izdelek podjetja Microsoft. Razvit je bil pod imenom Project Natal za igralno konzolo Xbox 360. Namenjen je igranju konzolnih iger in zato je narejen tako, da je dostopen čim širši množici uporabnikov. Pri izdelavi so

MB	ASRock Z77E-ITX
CPU	Intel Core i7 3770T 2.5 Ghz, 8MB L3 cache
GPU	EvGA Geforce GTX 650 1GB
RAM	Corsair vengeance 2 x 4 GB DDR3
SSD	Samsung 840 series 120 GB SSD
PSU	M4-ATX 6-30V DC/DC (250 Watt)

Tabela 3.1: Komponente računalnika v robotu.

Slika 3.3: ATRV mini.

uporabljene poceni komponente, kar pa se odraža v nenatančnosti senzorjev. Omogoča interakcijo z igralno konzolo ali računalnikom z uporabo gest in glasovno komunikacijo. Prva generacija je bila izdana leta 2010. Na sliki 3.4 je viden primerek prve generacije. Za komunikacijo med računalnikom in Kinectom je bilo razvitih kar nekaj odprtokodnih gonilnikov. To in nizka cena sta povzročila uporabo Kinecta v raziskovalne namene [38].

Kinect je podolgovata naprava, ki vsebuje barvno kamero, globinski senzor in več mikrofonov. Barvna kamera je 8-bitna z ločjivostjo 640x480 slikovnih elementov. Slike lahko zajema s frekvenco 9 - 30 Hz. Globinski senzor

ima prav tako ločljivost 640x480 slikovnih elementov, pri čemer je vsak slikovni element predstavljen z 11 biti. To omogoča 2048 stopenj občutljivosti za zaznavanje globine. Delovanje Kinecta je omejeno na razdaljo od 0.7 do 6 m, horizontalni vidni kot 57° in vertikalni vidni kot 43° [38, 39].

Slika 3.4: Kinect. Povzeto po [38].

3.4 Večsmerni sistem za zajemanje slik

Za zajemanje slik okolice robota smo uporabili sistem na sliki 3.5. Z večsmernim sistemom je mogoče zajeti 360° vidni kot glede na horizontalno ravnino. Sistem sestavljata večsmerno zrcalo in kamera. Zrcalo je lahko stožčasto, sferično, parabolično ali hiperbolično. Oblika zrcala določa, kako se bodo točke iz okolja preslikale na posneto sliko [40]. Zrcalo, uporabljeno na našem robotu, je vidno na sliki 3.5.

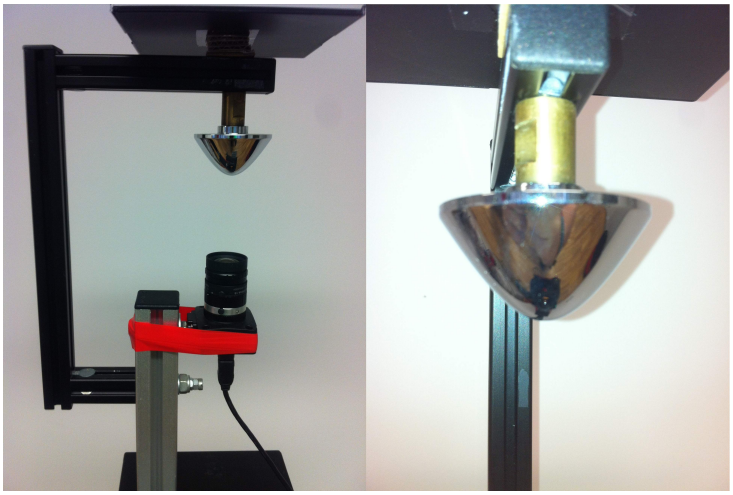
Kamera, ki smo jo uporabili na našem robotu, je izdelek podjetja Point Grey. Imenuje se Chameleon, njene specifikacije pa so vidne v tabeli 3.2. Upravljanje s kamero je mogoče preko priložene knjižnice FlyCapture.

3.5 Robotski operacijski sistem

Robotski operacijski sistem je odprtokodno orodje, ki omogoča razvoj programske opreme za različne robote. Izdan je pod licenco BSD, ki omogoča zastonj uporabo tako za komercialne kot za raziskovalne namene. Predstavlja zbirko orodij, knjižnic in pravil za enostavnejši razvoj robotov, ki izvajajo kompleksna, a hkrati robustna opravila. Roboti imajo ponavadi različno strojno opremo, kar onemogoči prenos programske kode. Potrebne prilagoditve zahtevjo poznavanje velikega števila sistemov in delovanja gonilnikov

Senzor	Sony ICX445 CCD, 1/3", 3.75 μm
Mega slikovnih elementov	1.3
Ločljivost	1296×964
Bitnost slik	8-bitne in 16-bitne
Format slik	Y8, Y16 (enobarvne),
	8-bitni in 16-bitni Raw Bayer data (barvne)
Upravljanje	FlyCapture SDK

Tabela 3.2: Specifikacije kamere Chameleon CMLN-13S2C-CS.



Slika 3.5: Večsmerno zrcalo s kamero.

za strojno opremo. ROS-ov ekosistem vsebuje programsko opremo, ki skrbi za neodvisnost platforme in prenos sprogramirane programske opreme. Programiranje je omogočeno v treh jezikih: C++, Python in LISP.

Razvoj se je začel leta 2007 na Stanfordu v laboratoriju za umetno inteligenco. Jeseni tega leta je razvoj prevzelo podjetje Willow Garage. Do sedaj je bilo izdanih že osem različic. Zadnja je bila izdana 22. julija 2014 pod imenom Indigo Igloo [41]. Naš sistem je zasnovan na sedmi različici imenovani Hydro Medusa.

Razvoj zelo kompleksnih robotov vključuje sodelovanje velike množice

ljudi, kjer je vsak strokovnjak za svoje področje. Robot pri izvajanju nalog, ki se ljudem zdijo trivialne, potrebuje veliko množico pravil in navodil za uspešno opravljanje naloge. Za lažje združevanje delov programske opreme je ROS narejen tako, da deluje karseda distribuirano in modularno. Tako lahko uporabniki ROS-a sami izberejo, katere že narejene komponente bodo uporabili in katere bodo implementirali sami.

Modularna zasnova operacijskega sistema je pripomogla k nastanku velike skupnosti razvijalcev programske opreme. Skupnost prispeva veliko repozitorijev z velikim številom paketov, ki vsebujejo implementacije najrazličnejših algoritmov s področja robotike, računalniškega vida, itd. Edini problem pri tako velikem številu prispevkov je, da marsikateri del programske kode ni dobro dokumentiran, kar otežuje uporabo [42, 43, 44].

Zgradba sistema ROS je v poglavjih, ki sledijo, opisana tako, kot je predstavljena na uradnih straneh [41] in članku [43].

3.5.1 Glavni koncepti

ROS ima modularnost implementirano kot računski graf *(ang. Computation Graph)*. To je P2P *(ang. Peer to Peer)* omrežje procesov, ki skupaj procesirajo podatke. Prednost take zgradbe sistema je dinamičnost priključitve novega procesa v staro zgradbo. Glavni koncepti, ki so opisani v nadaljevanju, so vidni na sliki 3.6.

- **Vozlišče** *(ang.* ***Node****)* procesira podatke. Sistem ponavadi vključuje veliko število vozlišč, pri čemer je vsako zadolženo za nalogo manjše težavnosti. Naloge so ponavadi funkcionalno zaključene, zato da imamo na logičnem nivoju lepo organiziranost. Naprimer v sistemu imamo eno vozlišče, ki sprejema in procesira podatke iz Kinecta, drugo vozlišče je zadolženo za lokalizacijo robota na podlagi procesiranih podatkov iz Kinecta, tretje je zadolženo za planiranje naslednjega premika robota, itd.

- **Glavno vozlišče** *(ang.* ***Master****)* je zadolženo za spremljanje aktiv-

nih vozlišč. Ob zagonu novega vozlišča se le-to registrira v glavnem vozlišču. Glavno vozlišče si shrani podatke o novem vozlišču. Glavno vozlišče omogoča pošiljanje sporočil in uporabo storitev. Ko želi kakšno vozlišče komunicirati z drugim, ga lahko poišče preko glavnega vozlišča. Tako je možno dinamično po potrebi ustvarjanje povezav med vozlišči. Povezave se vzpostavljajo neposredno med vozlišči.

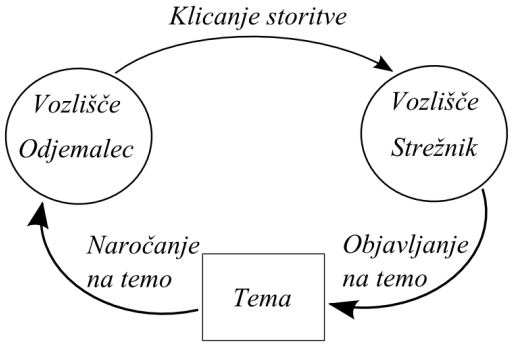
- **Sporočila** *(ang.* ***Messages****)* vsebujejo podatke, ki jih želi eno vozlišče prenesti drugemu. To je podatkovna struktura, ki vsebuje različne tipe podatkov. Vsebuje lahko primitivne tipe, kot so cela števila, števila s plavajočo vejico, nizi, itd., in celo kompleksnejše podatke, kot so druga sporočila. Kompleksnejša struktura je lahko vgnezdena poljubno globoko.

- **Teme (ang. Topics)** so del transportnega sistema, ki deluje po sistemu objavi-naroči. Vozlišče objavi sporočilo na neki temi. Vsa ostala vozlišča, ki bi rada dobila njegove informacije se naročijo na temo in transportni sistem poskrbi, da vsa vozlišča dobijo enako sporočilo. Vozlišč, ki na neko temo objavljajo podatke, je lahko več. Nobeno vozlišče, ki je povezano na neko temo, načeloma ne pozna drugih, ki so prav tako povezana nanjo. Število tem, na katere so lahko vozlišča povezana, ni omejeno.

- **Storitve** *(ang.* ***Services****)* omogočajo sinhrone transakcije med dvema vozliščema, kar s temami ni mogoče enostavo zagotoviti. Primerne so za komunikacijo, kjer eno vozlišče zahteva neko točno določeno storitev od drugega. Definirane so s točno določenim imenom in parom sporočil. Eno sporočilo predstavlja zahtevo in drugo odgovor. Celoten sistem deluje po principu odjemalec/strežnik. Vozlišče (odjemalec), ki želi izvedbo neke storitve, pošlje zahtevo vozlišču, ki storitev izvaja (strežnik) in počaka na odgovor. Strežnik zahtevo obdela in pošlje odgovor odjemalcu. Ta princip je analogen spletnim storitvam.

- **Vreče** *(ang.* ***Bags****)* so format za shranjevanje in ponovno pošiljanje

sporočil na teme, iz katerih je bil original prebran in shranjen. To je še posebej uporabno v fazi razvoja sistema, ko nujno potrebujemo podatke za testiranje. Vsakokratno zbiranje podatkov s pomočjo senzorjev je dolgotrajno in nepotrebno. S tem konceptom se izognemo nepotrebnemu zapravljanju časa in omogočimo ponovljivost testov.



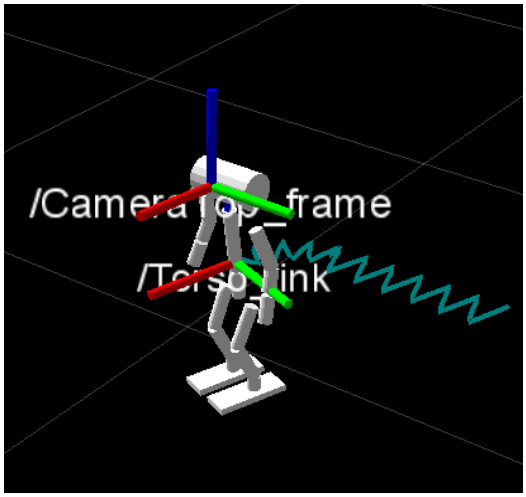
Slika 3.6: ROS-ovi glavni koncepti.

3.5.2 Višjenivojski koncepti

Transformacije in koordinatni sistemi

Robotski sistem mora slediti različnim prostorskim odnosom. Pri lokalizaciji robota s Kinectom je naprimer potrebna pretvorba podatkov iz koordinatnega sistema globinske kamere v koordinatni sistem robota. Na sliki 3.7 sta vidna dva izmed mnogih koordinatnih sistemov robota. Robot s takim številom sklepov ima v vsakem sklepu svoj koordinatni sistem. Tako je mogoče enostavno slediti premikom okončin robota. Za enostavno sledenje koordinatnim sistemom skozi čas, je v ROS-u narejen transformacijski sistem imenovan *tf*. Koordinatni sistemi so med seboj odvisni, zato so organizirani v drevesno strukturo. Sistem omogoča avtomatsko pretvorbo med koordinatnimi sistemi, kar olajša programiranje, saj ni treba pretvorb delati na roke. Ker sistem sledi koordinatnim sistemom skozi čas, je možna pretvorba v koordinatni sistem v točno določenem času, tudi že preteklem. Sistem *tf*

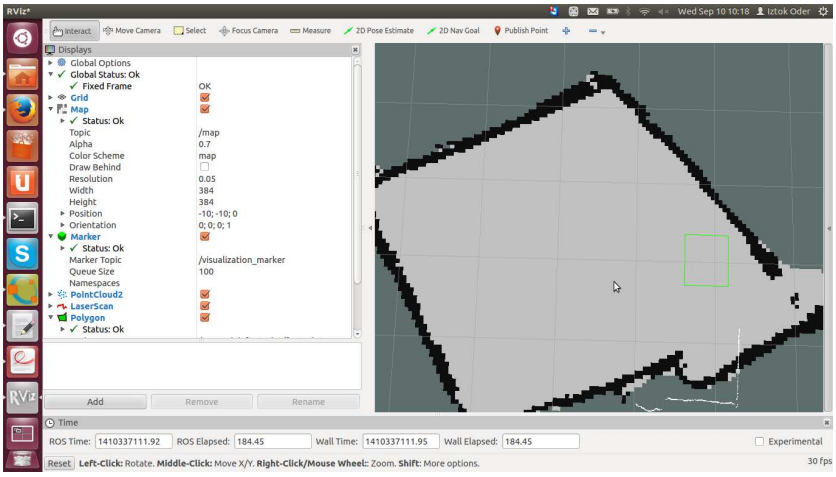
lahko deluje tudi v porazdeljenem sistemu. To pomeni, da so vse informacije o koordinatnih sistemih na voljo vsem komponentam ROS-a, na kateremkoli računalniku, ki ga porazdeljeni sistem uporablja.



Slika 3.7: Koordinatni sistemi robota. Povzeto po [41].

Vizualizacija in nadzorovanje

Med razvojem robotske programske opreme je velikokrat potrebno opazovati obnašanje sistema. Izpisovanje stanja v terminalu je najpogostejša tehnika razvijalcev za razhroščevanje. Napake je iz takega izpisa včasih težko razbrati. Za lažje in hitrejše odpravljanje napak potrebujemo bolj vizualni tip informacije. Sama zasnova ROS-a omogoča razvoj programov, ki nam vizualno prikažejo podatke, ki jih vozlišča objavljajo na temah. Eden takih programov je *rviz*, ki je že del vsake distribucije ROS-a. Omogoča vizualizacijo najrazličnejših tipov sporočil, ki so že del ROS-a. Poleg tega omogoča tudi povratno komunikacijo z vozlišči preko nekaj gumbov. Na sliki 3.8 je na levi strani viden bogat nabor sporočil, ki jih lahko vizualiziramo. Vizualizacija se naredi v sosednjem ali v novem oknu. Na sliki sta vidna vizualizacija mape in robota v *rvizu*.



Slika 3.8: *rviz*.

3.6 Implementacija programskega dela sistema

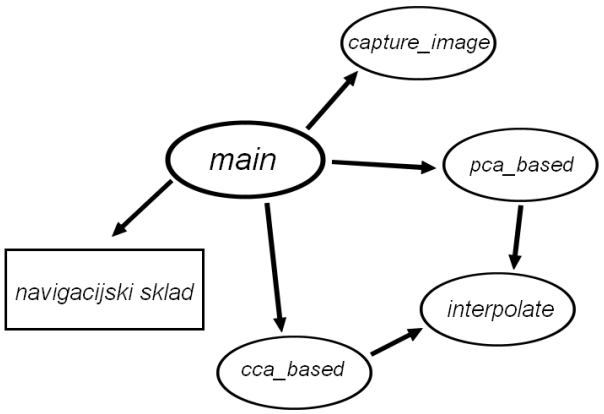
Programski del sistema je razdeljen na pet delov. V ROS-u to predstavlja pet vozlišč, ki so zadolžena za različne naloge. Zaradi specifičnosti naloge, ki jo mora sistem opravljati, se štiri vozlišča obnašajo kot storitev. Poleg njih imamo še eno glavno vozlišče, ki vse skupaj povezuje v zaključeno celoto. Na sliki 3.9 je razvidna razdelitev vozlišč. Vozlišča so razdeljena in opravljajo naloge po naslednjem sistemu:

- Primarna naloga glavnega vozlišča je komunikacija z uporabnikom sistema. Omogoča mu testiranje lokalizacijskih metod in snemanje učne množice. V primeru, da želi uporabnik posneti učno množico, lahko to stori avtomatično z izbiro točk v *rvizu* ali pa ročno z uporabo igralnega ploščka in tipkovnice. V primeru testiranja ali avtomatičnega snemanja učne množice se vozlišče opre na takoimenovan navigacijski sklad, ki pripelje robota s pomočjo Kinecta na izbrano lokacijo. Lokacijo, kamor želimo poslati robota, lahko določimo s kliki v *rvizu*. Ko testiramo metode za lokalizacijo, vozlišče zaprosi druga vozlišča za panoramsko sliko trenutne lokacije in lokalizacijo na podlagi te slike. Po pridobitvi predvidenih pozicij te informacije izriše v *rvizu*.

- Drugo vozlišče skrbi za komunikacijo s kamero. Imenuje se *capture_image*. Nastavlja parametre za zajem slike, kot so odprtost zaslonke, osvetljenost, tip formata slike, itd. Glavnemu vozlišču ponuja zajem slike na ukaz. Poleg samega slikanja opravi tudi pomembno transformacijo originalne barvne slike v sivinsko sliko in transformacijo iz polarnega koordinatnega sistema v kartezičnega.

- Tretje in četrto vozlišče sta zadolžena za lokalizacijo. Imenujeta se *cca_based* in *pca_based*. Obe vozlišči najprej iz učnih podatkov, s pomočjo metod PCA, KPCA, CCA in KCCA (poglavje 2), pripravita parametre za lokalizacijo. Ko glavno vozlišče pošlje sliko, izračunata predvideno pozicijo robota in jo posredujeta glavnemu vozlišču. Pri izračunu uporabljata peto vozlišče, ki je zadolženo za interpolacijo koeficientov, ki mu jih pošljeta, in natančnejšo lokalizacijo na podlagi teh informacij.

- Peto vozlišče izvaja baricentrično interpolacijo in izračun pozicijskih podatkov. Imenuje se *interpolate* in opravlja funkcijo storitve. Na voljo je tretjemu in četrtemu vozlišču.



Slika 3.9: Grafični prikaz delovanja vseh vozlišč. Smer puščice označuje uporabo storitev. Npr. *cca_based* uporablja storitev vozlišča *interpolate*.

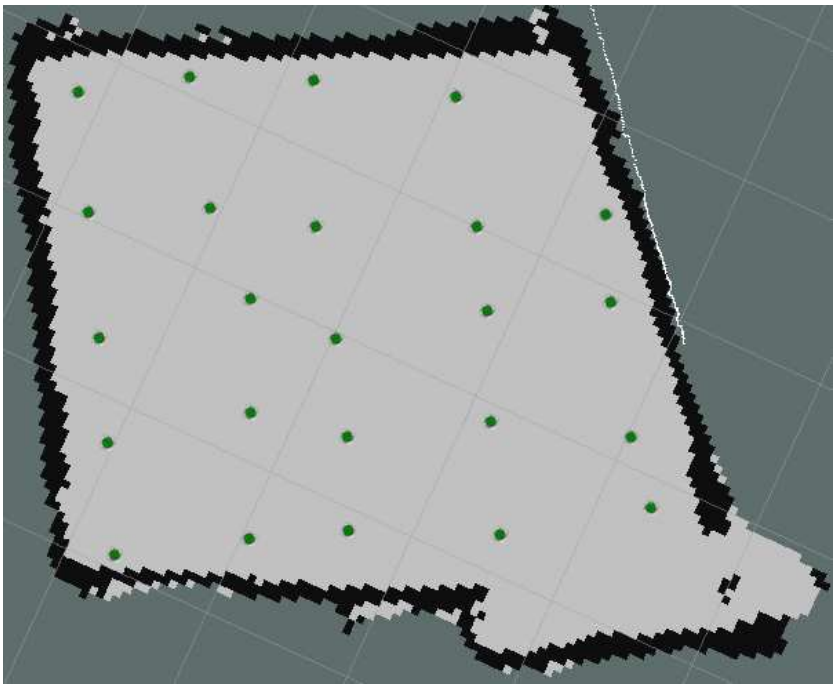
Poglavje 4

Eksperimentalni rezultati

4.1 Zajemanje podatkov

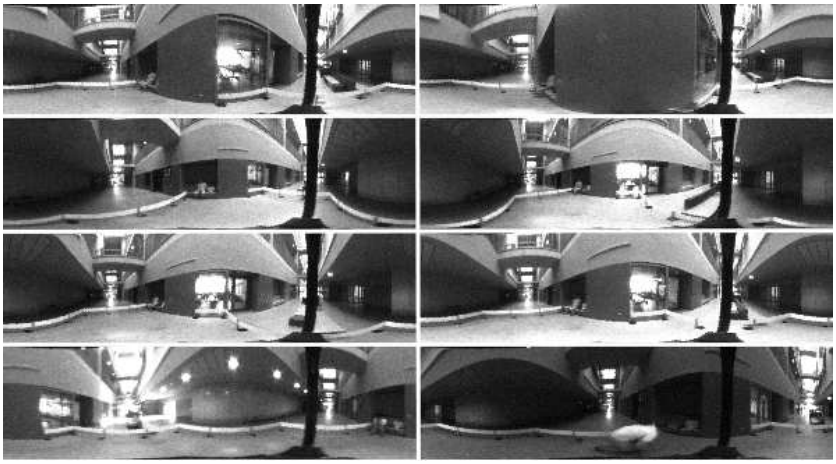
Z robotom smo na območju velikem približno $3.5m \times 4m$, vidnem na sliki 4.3, posneli zemljevid zasedenosti s pomočjo Kinecta, kot se vidi na sliki 4.1. Robota smo ročno vozili naokoli in pri tem pazili, da se je zemljevid pravilno zgradil. Zemljevid nam je služil kot referenčni koordinatni sistem. Na istem območju smo posneli 24 učnih slik na različnih razmakih med $40cm$ in $60cm$, kot je vidno na sliki 4.2. Za vsako sliko smo pridobili tudi lokacijo, kjer je bila posneta, s pomočjo zemljevida zasedenosti in Kinecta. Kinect je precej nenatančen, zato tudi lokacije niso čisto natančne. Posneli smo slike velikosti 1296×964 slikovnih elementov, ki smo jih pretvorili v panoramske slike velikosti 82×298 slikovnih elementov. Za učno množico smo vsako sliko najprej zarotirali tako, da je bil kot rotacije 0, nato pa smo jo zarotirali 50-krat (rotacijski interval je bil širok $7.20°$) in tako dobili 1200 slik. Na istem območju smo posneli tudi testne slike na drugih lokacijah in pod različnimi rotacijami. Nekaj posnetih slik se vidi na sliki 4.4. Zgornjih šest slik je del učne množice. Spodnji dve sta testni sliki. Pri zajemanju slik osvetljenosti ni bilo mogoče čisto nadzorovati, zato je opazna majhna razlika med njimi. V testnih slikah je mogoče opaziti tudi nekaj ovir, kot so naprimer ljudje.

Slika 4.1: Robot pri snemanju zemljevida zasedenosti.



Slika 4.2: Mapa z lokacijami učnih slik.

Slika 4.3: Ograjeno območje, kjer smo posneli slike.



Slika 4.4: Panoramske slike posnete na območju na sliki 4.3.

4.2 Rezultati lokalizacije

V poglavjih, ki sledijo, bodo predstavljeni rezultati lokalizacije na slikah posnetih, kot piše v poglavju 4.1. Pri vseh metodah, kjer je bila za natančno lokalizacijo uporabljena interpolacija vektorjev koeficientov učnih slik, je bila

le-ta narejena na mreži $1cm \times 1cm$. Za primerjavo med natančimi lokacijami in predvidenimi z metodami smo uporabili evklidsko razdaljo. Povprečno napako kotov smo izračunali kot povprečje absolutnih razlik med napovedanimi in točnimi koti. Koti niso nikjer vizualizirani zaradi premajhne povprečne napake.

4.2.1 Lokalizacija s PCA

Pri lokalizaciji s PCA smo uporabili različne pristope, za računanje podprostorov in projeciranje testnih slik nanje. Podprostore smo računali na dva načina.

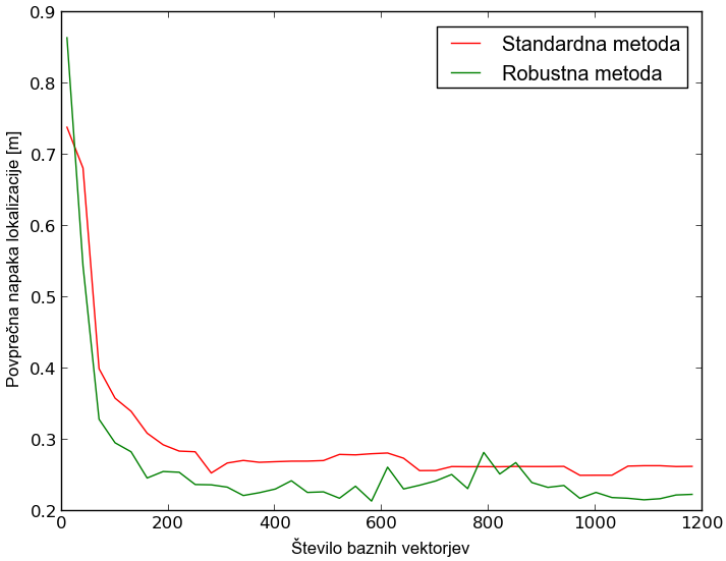
1. Pri prvem smo izračunali bazne vektorje enega podprostora na množici vseh 1200 slik. Na ta podprostor smo projecirali učne slike z navadno in robustno metodo. Obe metodi sta predstavljeni v poglavju 2.3.1.

2. Pri drugem načinu smo množico 1200 slik razdelili na 50 podmnožic. V vsaki podmnožici je bilo 24 slik, ki so bile enako orientirane. Na vsaki od teh podmnožic smo izračunali PCA in tako dobili 50 podprostorov s 24 baznimi vektorji. Na dobljene podprostore smo testne slike projecirali samo z robustno metodo.

Lokalizacija z enim podprostorom

V prvem poskusu smo izračunali podprostor s prvim pristopom. Dobili smo 1200 baznih vektorjev, ki razpenjajo ta podprostor. Vseh vektorjev ne potrebujemo. Preveliko število baznih vektorjev lahko povzroči preveliko prilagajanje testnih slik učni množici, premajhno pa vsebuje premalo informacije. V obeh primerih je rezultat nenatančna lokalizacija.

Da bi videli, koliko vektorjev je potrebnih za karseda natančno lokalizacijo, smo najprej testirali, kako se napaka pri lokalizaciji spreminja v odvisnosti od števila vektorjev. Najmanjše število vektorjev, ki smo jih vzeli, je bilo 10, največje pa 1180. Za projeciranje učnih slik v podprostor smo uporabili standardno metodo.

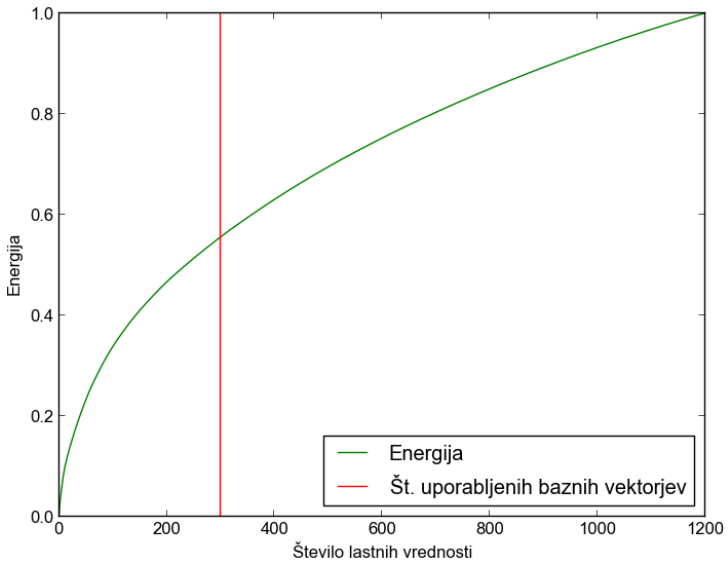


Slika 4.5: Slika prikazuje povprečno napako lokalizacije v odvisnosti od števila baznih vektorjev pri uporabi enega podprostora. Rdeča črta ponazarja lokalizacijo s standardno metodo, zelena pa z robustno metodo.

Iz slike 4.5 je razvidno, kako se povprečna napaka spreminja v odvisnosti od števila baznih vektorjev. Najmanjša napaka standardne metode je bila pri uporabi 970 baznih vektorjev in je znašala 25cm. Robustna metoda je imela pri uporabi 580 vektorjev napako 21.4cm. Lokalizacija je bila natančnejša pri uporabi robustne metode v večini primerov. Večji skoki napake, kot jih lahko opazimo pri robustni metodi, so posledica nestabilnosti zaradi naključnosti izbire slikovnih elementov.

Iz slike je razvidno tudi, da se napaka ne spreminja za več kot nekaj centimetrov nekje od vektorja številka 300 naprej. Iz tega lahko sklepamo, da več kot 300 baznih vektorjev ne potrebujemo. Podobno lahko sklepamo iz opazovanja energije, ki jo vsebujejo lastni vektorji. Slika 4.6 prikazuje energijo. Pri uporabi 300 baznih vektorjev dobimo povprečno napako lokalizacije s standardno metodo 25.5cm in 5.2°. Povprečna napaka lokalizacije z robustno metodo je 23.3cm in 4.8°.

Slika 4.7 prikazuje napovedane lokacije za vsako testno sliko. Črni krogci



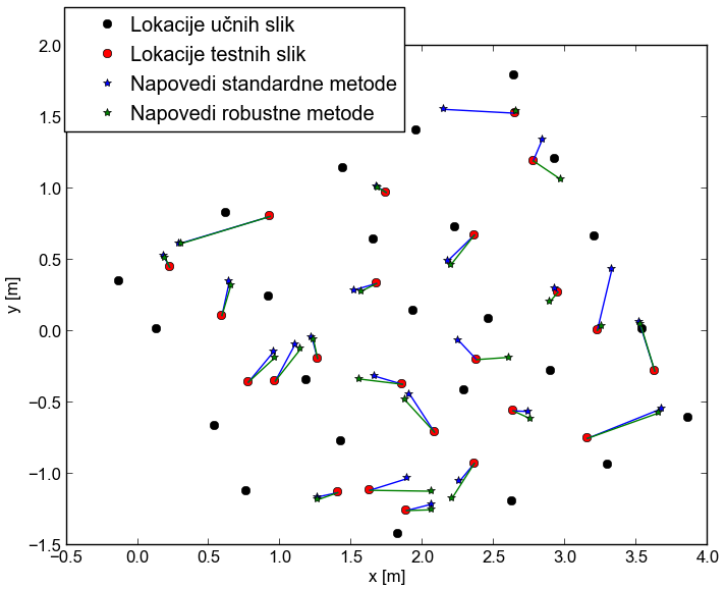
Slika 4.6: Zelena krivulja prikazuje energijo baznih vektorjev.

predstavljajo lokacije učnih slik, rdeči pa točne lokacije testnih slik. Z modro barvo so označene lokacije napovedane s standardno metodo, z zeleno pa lokacije napovedane z robustno metodo.

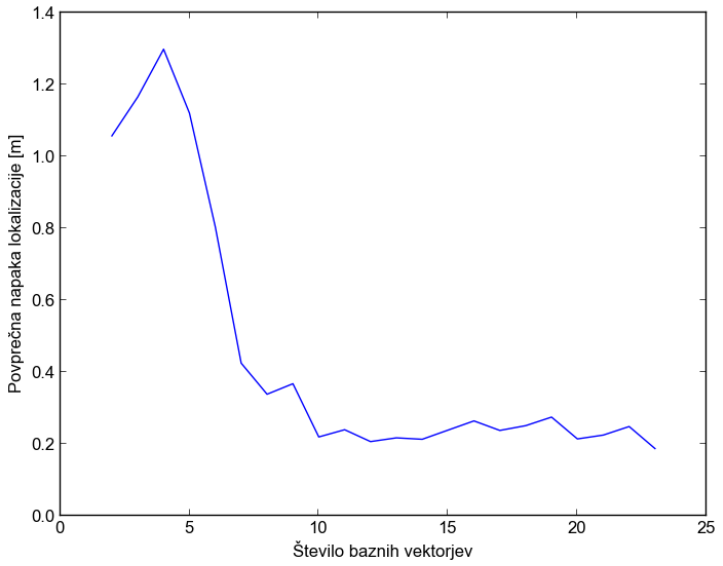
Lokalizacija z večimi podprostori

Za drugi poskus smo uporabili drugi pristop. Dobili smo 50 podprostorov, pri čemer je bil vsak razpet z 24 baznimi vektorji. Za projeciranje učnih slik smo uporabili standardno metodo.

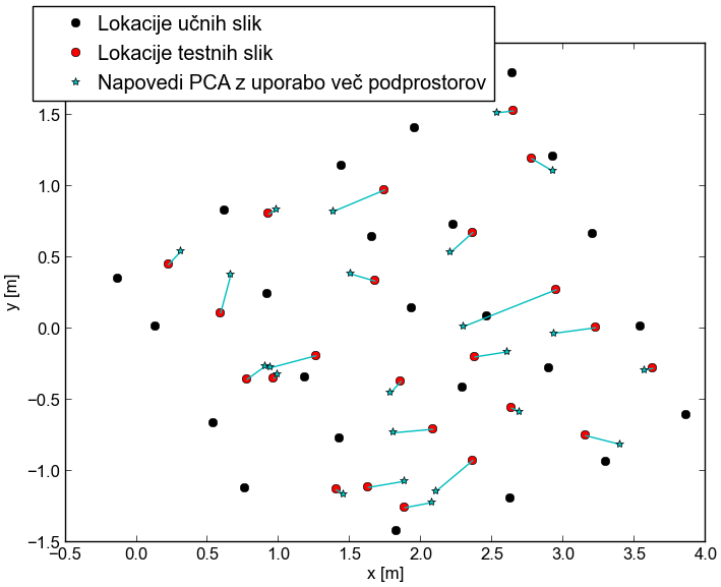
Za določitev števila potrebnih baznih vektorjev smo izračunali povprečno napako lokalizacije v odvisnosti od števila vektorjev, kot se to vidi na sliki 4.8. Najmanjša povprečna napaka 16.2cm je bila pri uporabi 21 vektorjev. Iz slike je razvidno, da napaka s številom vektorjev pada. Pri številu vektorjev 11 se umiri in je povprečna napaka podobna napaki robustne metode. Velika je 22.7cm Za zmanjšanje prostorskih zahtev, je torej število potrebnih vektorjev 11. Povprečna napaka kota je bila pri uporabi 11 vektorjev 5.7°. Slika 4.9 prikazuje predvidene lokacije, ki so označene z modrimi zvezdami.



Slika 4.7: Slika prikazuje napovedane lokacije testnih slik z uporabo enega podprostora.



Slika 4.8: Slika prikazuje povprečno napako lokalizacije v odvisnosti od števila baznih vektorjev pri uporabi 50 podprostorov.



Slika 4.9: Slika prikazuje napovedane lokacije z modro zvezdo. Pri tem testu je bilo uporabljenih 50 podprostorov, za vsako rotacijo slike svoj.

Komentar

Lokalizacija s PCA se je izkazala za izjemno uporabno. Pri uporabi enega podprostora za predstavitev vseh rotacij slik sta se tako standardna metoda kot robustna metoda projekcije v podprostor izkazali za precej dobro. Robustna metoda je delovala malce bolje predvsem zaradi določenih ovir, ki so se pojavile v testnih slikah. Teh ni bilo veliko, zato je tudi standardna metoda delovala precej dobro.

Pri uporabi 50 podprostorov (za vsako rotacijo svojega), je bila napaka lokalizacije podobna kot pri uporabi enega podprostora. Pri uporabi manjšega števila baznih vektorjev, je bila napaka približno enaka robustni metodi. Pri uporabi večjega števila vektorjev, pa je celo padla pod $20cm$. To se pri uporabi samo enega podprostora ni zgodilo.

Povprečna napaka kota je bila neglede na število podprostorov zelo majhna. Prav tako ovire in osvetljenost niso predstavljale večjih problemov.

4.2.2 Lokalizacija s KPCA

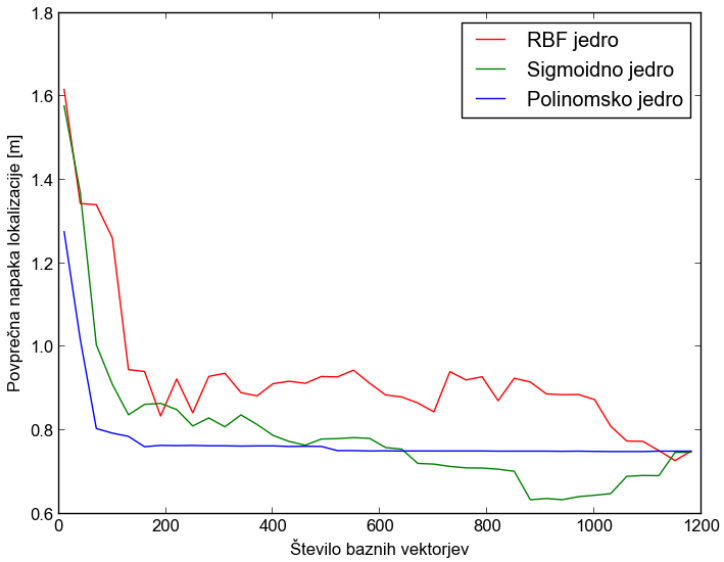
Pri KPCA je bilo potrebno najprej določiti, katero jedro bomo uporabili in njegove parametre. Poskuse smo izvedli na polinomskem, RBF in sigmoidnem jedru. Izkazalo se je, da vsi trije dajejo podobne rezultate. Povprečna napaka je bila pri vseh okoli $75cm$ pri uporabi vseh 1200 vektorjev. Ta napaka je precej večja kot pri navadni PCA.

Pri KPCA ponavadi vzamemo vse bazne vektorje prav zaradi visoke dimenzionalnosti podatkov. Da bi ugotovili, če preveliko število baznih vektorjev vpliva na preveliko prilagajanje učni množici, smo naredili test za ugotavljanje odvisnosti napake od števila baznih vektorjev. Uporabili smo RBF jedro s širino 40, polinomsko jedro s stopnjo 2 in sigmoidno jedro z naklonom 4 in zamikom 0. Na sliki 4.10 se vidijo odvisnosti povprečne napake od števila vektorjev za vsakega od jedr.

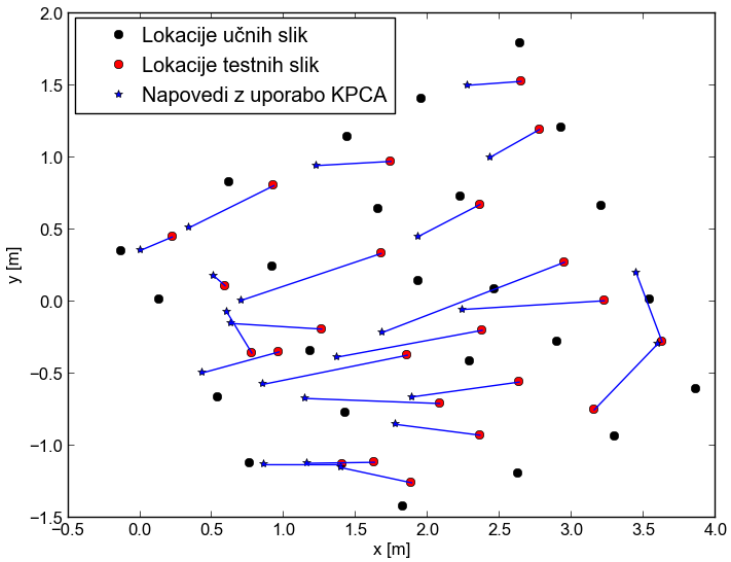
Lokalizacija z RBF jedrom je zelo odvisna od števila vektorjev. Potrebujemo namreč vse za enako natančnost kot pri sigmoidnem in polinomskem. Pri uporabi polinomskega jedra potrebujemo samo 170 baznih vektorjev. Napaka se pri večjem številu baznih vektorjev praktično ne spreminja. Najboljši rezultati so se pokazali pri uporabi sigmoidnega jedra in 900 baznih vektorjev. Povprečna napaka se je tam spustila celo do $63cm$. Povprečna napaka kota je bila $9.9°$. Napovedi pri uporabi sigmoidnega jedra so vidne na sliki 4.11.

Komentar

Lokalizacija z metodo KPCA se je izkazala za neprimerno. Transformacija slik v višjedimenzijski prostor ne naredi ločnice med slikami, temveč jo zabriše. To vpliva na primerjavo vektorjev koeficientov projekcij, saj se le-ta naredi z napačnimi vektorji koeficientov.



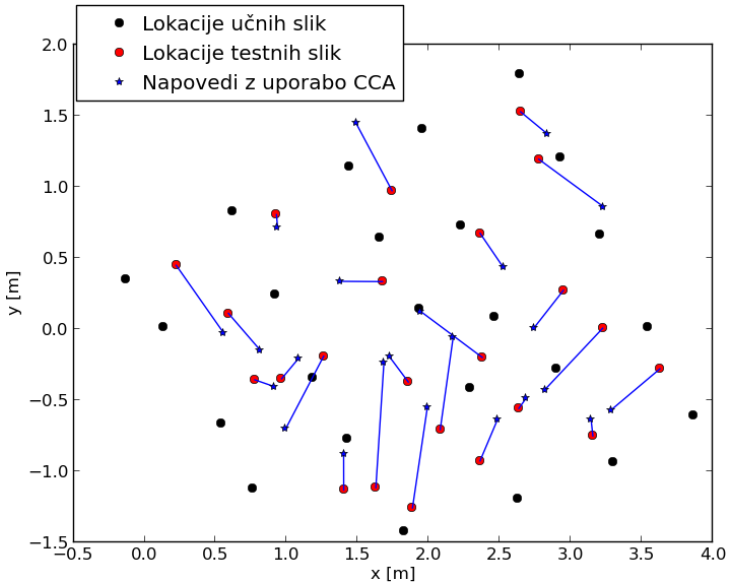
Slika 4.10: Slika prikazuje povprečno napako lokalizacije z metodo KPCA v odvisnosti od števila baznih vektorjev. Rdeča krivulja prikazuje napako pri uporabi RBF jedra, zelena napako pri uporabi sigmoidnega jedra in modra napako pri uporabi polinomskega jedra.



Slika 4.11: Slika prikazuje predvidene lokacije pri uporabi KPCA s sigmoidnim jedrom. Modre zvezde označujejo predvidene lokacije.

4.2.3 Lokalizacija s CCA

Pri CCA je potrebna uporaba množice slik in množice njihovih pozicij. Dimenzija obeh koordinatnih sistemov, ki jih pri tem dobimo, je največ 3. Uporabili smo učno množico slik, kjer je vsaka slika zarotirana 50-krat. Glede na to, da zna CCA najti ločnico samo med linearno odvisnimi podatki, nas je zanimalo, če uporaba kota v množici pozicij vpliva na lokalizacijsko točnost. Izkazalo se je, da neglede na ta podatek dobimo enako lokalizacijsko točnost. Kota iz tega razloga ni mogoče natančno napovedati. Povprečna napaka kota je bila kar 110.2° Povprečna napaka lokalizacije je bila 39.2cm. Napovedane pozicije robota so vidne na sliki 4.12.



Slika 4.12: Slika prikazuje napovedane lokacije z uporabo CCA. Modre zvezde prikazujejo napovedi.

Komentar

CCA se kljub privlačnosti, zaradi majhnega števila podatkov, ki jih je potrebno shraniti v pomnilniku, ni izkazala za uporabno pri lokalizaciji. Vektorji, ki jih pridobimo vsebujejo premalo informacije za natančno napove-

dovanje lokacije robota. Pri istem številu baznih vektorjev se PCA izkaže slabše, vendar imamo tam možnost vzeti večje število vektorjev in tako precej izboljšati rezultat.

4.2.4 Lokalizacija s KCCA

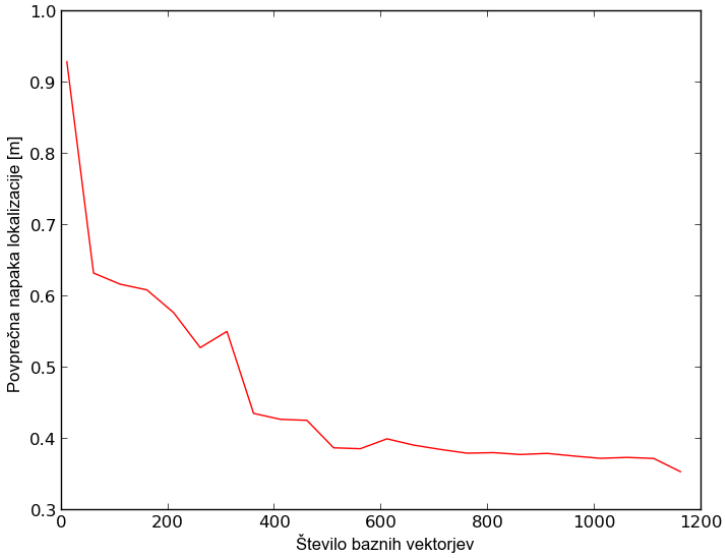
KPCA se je slabo obnesel pri uporabi kateregakoli jedra. Zato smo pri KCCA uporabili jedro samo na množici znanih pozicij učnih slik. Preizkusili smo polinomsko, RBF in sigmoidno jedro. Vsa jedra so, pri uporabi vseh 1200 vektorjev, dala podobne rezultate. Povprečna napaka se je, pri uporabi različnih parametrov, gibala med $36cm$ in $40cm$. Za natančnejšo analizo smo izbrali RBF jedro, ker je imelo najmanjšo povprečno napako. Testirali smo, kako se napaka spreminja, pri različnih širinah jedra. Testirali smo na širinah med 5 in 95. Povprečna napaka se je prav tako gibala med $35cm$ in $40cm$. Povprečna napaka kota rotacije se je gibala okoli 5°, kar je enako kot pri PCA.

Ugotoviti je bilo potrebno tudi kako se povprečna napaka lokalizacije spreminja v odvisnosti od števila vektorjev. To je pomembno za določitev števila potrebnih vektorjev za karseda natančno lokalizacijo. Testi so pokazali, da je dovolj, če uporabimo samo polovico vseh vektorjev, kot se vidi na sliki 4.13. Uporabljeno je bilo RBF jedro s širino 15.

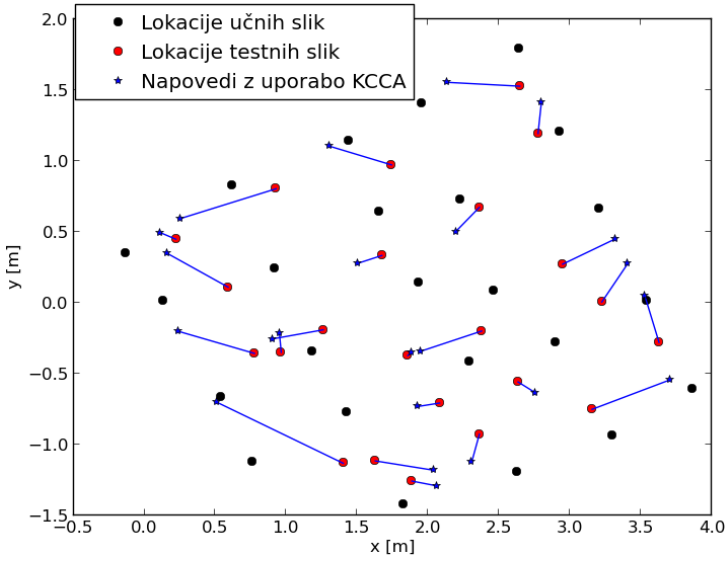
Najmanjša povprečna lokalizacijska napaka je bila $35.7cm$. Povprečna napaka kota je bila v tem primeru 4.8°. Za lokalizacijo je dovolj, če uporabimo med 500 in 600 vektorji, ker se od tam naprej napaka ne spreminja več veliko. Napovedane lokacije na sliki 4.14 smo dobili z uporabo 500 vektorjev. Povprečna lokalizacijska napaka je bila $37.8cm$, napaka kota pa 5.4°.

Komentar

KCCA se v nasprotju s pričakovanji ni odrezala najbolje. Povprečna napaka kota rotacije je bila zadovoljivo majhna, medtem ko je bila napaka lokalizacije precej velika. Deloma je za to kriva osvetljenost slik, deloma pa ovire, ki so se pojavile v testnih slikah.



Slika 4.13: Slika prikazuje povprečno napako lokalizacije z metodo KCCA v odvisnosti od števila baznih vektorjev.



Slika 4.14: Slika prikazuje napovedane lokacije z uporabo KCCA. Modre zvezde prikazujejo napovedi.

4.2.5 Primerjava rezultatov

Za večjo preglednost so vsi rezultati zbrani v tabeli 4.1. V stolpcu **Metoda** so imena metod, s katerimi smo izvajali lokalizacijo. PCA_1prostor_std označuje uporabo PCA za izračun enega podprostora in uporabo standardne metode za projekcijo testnih slik. PCA_1prostor_rob označuje uporabo PCA za izračun enega podprostora in uporabo robustne metode za projekcijo testnih slik.PCA_50prostor označuje uporabo PCA za izračun 50 podprostorov in uporabo robustne metode za projekcijo testnih slik.

Metoda	**Povprečna napaka lokalizacije** $[cm]$	**Povprečna napaka kota** $[^\circ]$
PCA_1prostor_std	25.5	5.2
PCA_1prostor_rob	23.3	4.8
PCA_50prostor	22.7	5.7
KPCA	63.6	9.9
CCA	39.2	110.2
KCCA	37.8	5.7

Tabela 4.1: Rezultati povprečnih napak napovedi lokalizacije.

Iz tabele 4.1 je razvidno, da so se najbolje odrezale metode, ki temeljijo na PCA. Povprečna napaka kota je bila pri uporabi teh metod zelo majhna in se je gibala med 4° in 6°. Povprečna napaka lokalizacije gibala med 22 in $26cm$, kar je relativno malo, če gledamo velikost poligona, ki je znašala $3.5m \times 4m$. Najbolje se je sicer odrezala PCA s 50 podprostori, vendar so si rezultati vseh treh metod precej podobni.

Preostale tri metode so se odrezale precej slabše. KPCA je imela največjo povprečno lokalizacijsko napako, ki je znašala $63.6cm$. To je skoraj trikrat več kot pri najboljši metodi, ki temelji na PCA. KCCA in CCA sta imeli zelo podobno povprečno lokalizacijsko napako. To nas je malo presenetilo, saj smo pričakovali, da bo povprečna napaka lokalizacije s KCCA bolj podobna povprečni napaki lokalizacije s PCA. Povprečna napaka kota dobljenega pri

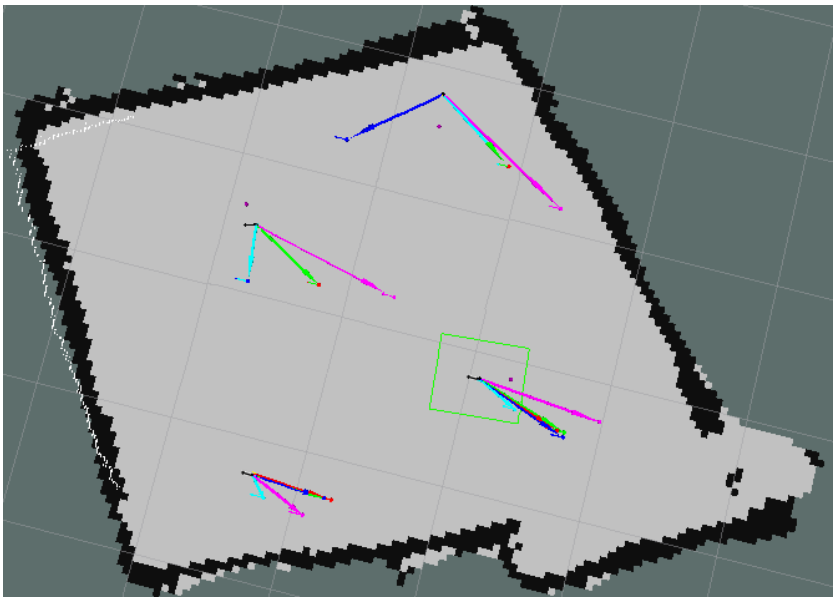
lokalizaciji s KCCA je bila zadovoljivo majhna, saj je znašala tako kot pri PCA s 50 podprostori le 5.7°. Povprečna napaka kota pri uporabi CCA pa je bila izjemno velika, saj je znašala 110.2°. V primerjavi s katero koli od ostalih metod je imela CCA daleč največjo povprečno napako kota. Povprečna napaka kota pri uporabi KPCA je bila večja kot pri uporabi PCA ali KCCA, vendar je bila vseeno relativno majhna, saj je znašala le 9.9°.

4.3 Sistem

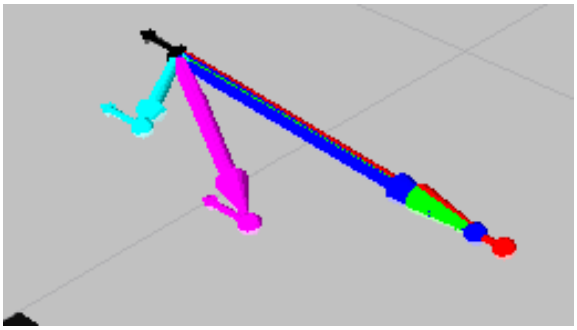
Sistem smo testirali tudi v interaktivnem načinu. Na zemljevidu, kot jo prikazuje slika 4.2, smo pognali robota. V *rvizu* smo določili na katere točke naj se robot premakne. Rotacijo, na katero se je zavrtel, smo določili naključno. Za lepši prikaz vizualnih rezultatov smo peljali robota samo na štiri lokacije. Poleg take predstavitve se na koncu izpišejo tudi napovedane pozicije.

Na slikah 4.15 in 4.16 se vidi prikaz v *rvizu*. Za lepši prikaz ni narisana napoved CCA, saj so njene puščice ponavadi zaradi nenatančnosti predolge, kar pa uniči preglednost. Na sliki 4.16 so prikazane puščice od blizu. Večje puščice, ki izhajajo iz črne krogle, kažejo na predvideno pozicijo, manjše pušice, pa predvideno rotacijo. Črna puščica prikazuje robotovo dejansko rotacijo. Marsikatera puščica se prekriva zaradi podobnih napovedi.

Pri poganjanju sistema sta se pojavila dva večja problem. Prvi problem, je natančnost Kinecta. Globinski senzor naj bi zaznaval razdalje do $6m$. V praksi se je izkazalo, da je pri razdaljah večjih od $4m$ natančnost zelo vprašljiva. V enem izmed prvih testov smo delali slike na zemljevidu velikosti približno $4m \times 6m$. Slike smo delali ročno, tako da smo vozili sami robota na željene pozicije. Pozicije, določene s pomočjo Kinecta, so imele napako veliko tudi več 10 centimetrov. Ta problem se je pojavil predvsem na sredini zemljevida, kjer ni bilo v bližini nobene referenčne točke. Zato smo zemljevid zmanšali na $3.5m \times 4m$. Pri tej velikosti se je problem zmanjšal. Ena izmed boljših možnosti bi bila seveda zamenjava Kinecta za kakšnen bolj natančen senzor, kot je naprimer LIDAR. Takšni senzorji, pa so neprimerno dražji od



Slika 4.15: Prikaz napovedanih pozicij robota na celotnem zemljevidu.



Slika 4.16: Prikaz ene izmed napovedanih pozicij robota.

Kinecta.

Naslednja težava, s katero smo se soočili, je bila osvetlitev prostora, v katerem smo snemali slike. Prostor je bil precej odprt, zato se je osvetlitev spreminjala glede na zunanje vremeske pogoje. Hitro se je lahko zgodilo, da smo na isti lokaciji dobili dve sliki, ki sta bili različno osvetljeni. To pa predstavlja ogromen probem pri lokalizaciji s katerokoli od uporabljenih metod. Če poskušamo lokalizirati robota s sliko, ki ima drugačno osvetlitev je napaka poljubno velika. Pri zajemanju slik je bilo zato potrebno ves čas paziti, da je količina svetlobe, ki pade skozi zaslonko kamere, približno enaka. Preostalo razliko smo delno odpravili z izravnavo histograma. Ta problem bi lahko odpravili tudi z večjim številom učnih slik, pri čemer bi isto lokacijo posneli pod različno jakostjo svetlobe. Učno množico bi tako izjemno napihnili, kar bi se odražalo v času izračuna baznih vektorjev in v času izračuna predvidene lokacije nove slike.

Poglavje 5

Zaključek

V diplomski nalogi smo se ukvarjali s problemom lokalizacije robota s pomočjo večsmerne kamere. Razvili smo sistem, ki omogoča ročno ali avtomatsko snemanje učnih slik in avtomatsko lokalizacijo na podlagi učnih slik. Za testiranje lokalizacijskih metod potrebujemo samo učne in testne slike. Sistem je zasnovan tako, da omogoča izvajanje testov, tako s pomočjo robota, kot tudi brez. Interaktivno poganjanje sistema smo izvajali na robotu ATRV mini, na katerega sta bila nameščena Kinect in večsmerna kamera. Za določanje referenčne pozicije je bil uporabljen Kinect in navigacijski sklad, ki je del ogrodja ROS.

Za lokalizacijo smo implementirali metode , ki temeljijo na podprostorih, pridobljenih s statističnimi metodami PCA, KPCA, CCA in KPCA. Ugotovili smo, da PCA najbolje predstavi slike v podprostoru, za namene lokalizacije. Najnatančnejša je bila prav lokalizacija s to metodo. Najmanjša povprečna napaka lokalizacije je bila $16.2cm$. Ostale metode se, zaradi nenatančnosti, niso izkazale primerne za lokalizacijo.

Možnih izboljšav je še zelo veliko. Izboljšali bi lahko metode za lokalizacijo. Pri CCA bi lahko naprimer raziskali, kako umetno povečati maksimalno dimenzijo podprostora. Potrebno bi bilo poiskati funkcije, s katerimi bi drugače predstavili kot rotacije in lokacijo, kot to naredi KCCA. Pomembna razlika pri tem je, da mora biti funkcija obrnljiva, zato da lahko napovedane

podatke, pridobljene z linearno regresijo, tudi interpretiramo.

Naslednji korak je kombiniranje napovedi različnih metod. Vsaka izmed metod za lokalizacijo je za neko določeno sliko najnatančneje napovedala lokacijo in rotacijo. Najočitnejša izboljšava je implementacija metode, s katero bi iz vseh napovedi izbrala le najboljšo. Tako bi lahko povprečno napako bistveno zmanjšali.

Sistema nismo preizkusili na prostem. Ker se zunaj svetlobni pogoji izjemno hitro spreminjajo, bi bila lokalizacija najverjetneje zelo nenatačna. Metode za lokalizacijo v trenutnem sistemu so namreč zelo občutljive na svetlobne pogoje. Potrebna bi bila implemetacija metode, ki bi odpravila problem različne osvetljenosti slik. Izravnava histograma je do neke mere opravila svojo nalogo, vendar bi se dalo ta problem z uporabo drugačnih metod še bolje rešiti. Prva možna rešitev je implementacija metode, ki bi avtomatsko nastavljala parametre kamere, kot je naprimer odprtost zaslonke, glede na svetlobne pogoje. Druga možnost je uporaba drugačnih vrst slik. V slikah bi lahko naprimer poiskali robove in jih uporabili za izračun podprostorov. Osvetljenost slike v tem primeru nebi imela nobenega vpliva.

Uporabnost sistema bi lahko povečali, če bi namesto globinskega senzorja, ki ga ima Kinect, uporabili kakšen drug senzor. Kinect ima efektivno natančnost določanja globine $3 - 4m$. Izvajanje eksperimentov v večjih prostorih je praktično nemogoče. Lokacije in rotacije robota, ki jih dobimo s pomočjo Kinecta, so v večjih prostorih popolnoma napačne. Od resničnih lahko odstopajo tudi nekaj 10 centimentrov. Namesto Kinecta bi bilo potrebno uporabiti natančnejši senzor, kot je naprimer LIDAR. Za določanje večje natančnosti referenčnih pozicij, pa so potrebni senzorji, ki niso poceni.

Največja nadgradnja sistema, ki bi jo lahko naredili, je implementacija navigacije po prostoru samo s pomočjo vizualne informacije. Lokacijo in orientacijo robota smo že precej natančno napovedali. Prvi predpogoj je torej izpolnjen. Implementirati bi bilo potrebno logiko za načrtovanje poti od začetne lokacije do končne. To še ni dovolj, saj bi se lahko robot vseeno zaradi kakšne napake pri lokalizaciji popolnoma zgubil. Potrebna bi bila tudi

implementacija logike za preverjanje pravilnosti vožnje po trenutni poti in ukrepanje v primeru napak. Tak sistem bi bil popolnoma avtonomen, saj bi tako za lokalizacijo kot za navigacijo uporabljal samo vizualno informacijo zajeto z večsmerno kamero. Predstavljeni pristop predstavlja pomemben korak pri razvoju takega sistema.

Literatura

[1] Wikipedia, "Global positioning system." http://en.wikipedia.org/wiki/Google_driverless_car. Obiskano 24.08.2014.

[2] Wikipedia, "Radar." http://en.wikipedia.org/wiki/Radar. Obiskano 24.08.2014.

[3] Wikipedia, "Google driverless car." http://en.wikipedia.org/wiki/Google_driverless_car. Obiskano 24.08.2014.

[4] S. Gibbs, "Google driverless car." http://www.theguardian.com/technology/2014/may/28/google-self-driving-car-how-does-it-work. Obiskano 24.08.2014.

[5] A. Reporter, "Google scores yet another win." http://www.arabiangazette.com/google-scores-win/. Obiskano 24.08.2014.

[6] Wikipedia, "Fisheye lens." http://en.wikipedia.org/wiki/Fisheye_lens. Obiskano 24.08.2014.

[7] Wikipedia, "Omnidirectional camera." http://en.wikipedia.org/wiki/Omnidirectional_camera. Obiskano 24.08.2014.

[8] R. Cipolla, D. Robertson, and B. Tordoff, "Image-based localisation," in *Proceedings of 10th International Conference on Virtual Systems and Multimedia*, pp. 22–29, 2004.

[9] T. Sattler, T. Weyand, B. Leibe, and L. Kobbelt, "Image retrieval for image-based localization revisited.," in *BMVC*, vol. 6, p. 7, 2012.

[10] W. Zhang and J. Kosecka, "Image based localization in urban environments," in *3D Data Processing, Visualization, and Transmission, Third International Symposium on*, pp. 33–40, IEEE, 2006.

[11] T. Sattler, B. Leibe, and L. Kobbelt, "Improving image-based localization by active correspondence search," in *Computer Vision–ECCV 2012*, pp. 752–765, Springer, 2012.

[12] C. Arth, M. Klopschitz, G. Reitmayr, and D. Schmalstieg, "Real-time self-localization from panoramic images on mobile devices," in *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pp. 37–46, IEEE, 2011.

[13] T. Sattler, B. Leibe, and L. Kobbelt, "Fast image-based localization using direct 2d-to-3d matching," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 667–674, IEEE, 2011.

[14] M. Jogan and A. Leonardis, "Robust localization using an omnidirectional appearance-based subspace model of environment," *Robotics and Autonomous Systems*, vol. 45, no. 1, pp. 51–72, 2003.

[15] D. Skočaj and A. Leonardis, "Appearance-based localization using cca," in *Computer vision - CVWW '04 : proceedings of the 9 th Computer Vision Winter Workshop*, pp. 205–214, Slovenian Pattern Recognition Society, 2004.

[16] W. K. Wong, W. ShenPua, C. K. Loo, and W. S. Lim, "A study of different unwarping methods for omnidirectional imaging," in *Signal and Image Processing Applications (ICSIPA), 2011 IEEE International Conference on*, pp. 433–438, IEEE, 2011.

[17] V. Grassi Junior and J. Okamoto Junior, "Development of an omnidirectional vision system," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 28, pp. 58 – 68, 03 2006.

[18] R. C. Gonzalez and R. E. Woods, *Digital image processing*. Prentice hall Upper Saddle River, NJ:, 2002.

[19] Wikipedia, "Histogram equalization." `http://en.wikipedia.org/wiki/Histogram_equalization`. Obiskano 19.08.2014.

[20] H. Hotelling, "Analysis of a complex of statistical variables into principal components.," *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[21] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[22] J. De Leeuw, "History of nonlinear principal component analysis," 2013.

[23] A. R. H. Swan and M. Sandilands, *Introduction to geological data analysis / A.R.H. Swan, M. Sandilands ; with statistical tables by P. McCabe*. Blackwell Science Oxford ; Cambridge, Mass., USA, 1995.

[24] J. Blasius and M. Greenacre, *Visualization and Verbalization of Data*. CRC Press, 2014.

[25] A. Leonardis and H. Bischof, "Robust recognition using eigenimages," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 99–118, 2000.

[26] D. Skočaj, *Robust subspace approaches to visual learning and recognition*. PhD thesis, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, Feb 2003.

[27] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," in *Artificial Neural Networks—ICANN'97*, pp. 583–588, Springer, 1997.

[28] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[29] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch, "Kernel pca and de-noising in feature spaces.," in *NIPS*, vol. 11, pp. 536–542, Citeseer, 1998.

[30] M. Borga, *Learning multidimensional signal processing*. PhD thesis, Linköping University, 1998.

[31] D. Hardoon, S. Szedmak, and J. Shawe-Taylor, "Canonical correlation analysis: An overview with application to learning methods," *Neural computation*, vol. 16, no. 12, pp. 2639–2664, 2004.

[32] M. Kuss and T. Graepel, "The geometry of kernel canonical correlation analysis," tech. rep., MPI-Technical Reports. URL http://www. kyb. mpg. de/publication. html, 2003.

[33] T. Melzer, M. Reiter, and H. Bischof, "Appearance models based on kernel canonical correlation analysis," *Pattern recognition*, vol. 36, no. 9, pp. 1961–1971, 2003.

[34] U. wiki, "10.2 sensors." `http://usarsim.sourceforge.net/wiki/index.php/10.2_Sensors`. Obiskano 9.09.2014.

[35] T. Pajdla and V. Hlaváč, "Zero phase representation of panoramic images for image based localization," in *Computer Analysis of Images and Patterns*, pp. 550–557, Springer, 1999.

[36] M. Jogan and A. Leonardis, "Panoramic eigenimages for spatial localisation," in *8-th International Conference on Computer Analysis of Images and Patterns*, pp. 558–567, Springer Berlin / Heidelberg, Sep 1999.

[37] M. Jogan and A. Leonardis, "Robust localization using eigenspace of spinning-images," in *IEEE Workshop on Omnidirectional Vision*, pp. 37–44, IEEE Computer Society, Jun 2000.

[38] Wikipedia, "Kinect." `http://en.wikipedia.org/wiki/Kinect`. Obiskano 22.08.2014.

[39] Microsoft, "Kinect." `http://msdn.microsoft.com/en-us/library/jj131033.aspx`. Obiskano 22.08.2014.

[40] S. K. Nayar, "Catadioptric omnidirectional camera," in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pp. 482–488, IEEE, 1997.

[41] ROS.org, "ROS." `http://wiki.ros.org/`. Obiskano 22.08.2014.

[42] ROS.org, "ROS." `http://www.ros.org`. Obiskano 22.08.2014.

[43] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, 2009.

[44] ROS.org, "ROS." `http://en.wikipedia.org/wiki/Robot_Operating_System`. Obiskano 22.08.2014.